Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт космических и информационных технологий
институт
Информатика
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ А.С. Кузнецов
подпись инициалы, фамилия
« 25 » 09 2017 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Программная система интерактивного управления системами жизнеобеспечения помещений
тема

Пояснительная записка

Руководитель _____ 22.09.2017 доцент, к.т.н. А.С. Кузнецов
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 22.09.2017 С.С. Баянов
подпись, дата инициалы, фамилия

Нормоконтролер _____ 22.09.2017 доцент, к.т.н. О. А. Антамошкин
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2017

РЕФЕРАТ

Выпускная квалификационная работа по теме «Программная система интерактивного управления системами жизнеобеспечения помещений» содержит 59 страниц текстового документа, 28 рисунков, 8 использованных источников.

Цель: разработать кроссплатформенное программное обеспечение для управления исполняющими модулями, позволяющее настраивать взаимодействие между программой и модулями с использованием библиотеки для исполняющих модулей, позволяющей связываться с программным обеспечением.

Во введении описывается обоснование разработки программной системы.

В первой главе описана аналитическая часть, содержащая сведения о средствах разработки и предметной области.

Во второй главе описывается процесс реализации программной системы, ее настройка и использование.

В заключении приводятся достигнутые результаты и перспективы.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире человек окружен различными электронными устройствами. Они помогают в работе, учёбе, в быту. Получили широкое применение в организации досуга. Человек стал доверять электронике управлять автомобилями, самолетами и даже собственной жизнью во время медицинских или спасательных операций. Электронные и роботизированные приборы стали обыденностью в наших домах.

Современные помещения оснащены инженерными системами, различающимися по своему составу и функциональному назначению компонентов. Среди них можно выделить технологические подсистемы, называемые системами жизнеобеспечения помещения:

- электроснабжение;
- вентиляция;
- кондиционирование;
- пароувлажнение;
- теплоснабжение;
- водоснабжение;
- канализация;
- противопожарные системы.
- охранные системы.

На волне доступности электронных цифровых устройств, стали появляться и системы “Умный дом”, которые можно условно разделить на 3 группы.

1) Коммерческие решения;

Предоставляют владельцу дома или квартиры возможность управлять различными модулями (чаще всего силовой нагрузкой). В качестве контроллера

подобного рода систем используется сервер, который по средствам Ethernet связывается с исполнительными модулями. Данный способ организации имеет несколько недостатков:

- Необходимость укладки дополнительных коммуникаций - прокладываются информационные линии для каждого из модулей, плюс в местах, где не имеется постоянное электроснабжение (светильники) - дополнительная линия питания (если имеется необходимость оставить стандартный способ включения/выключения света). Если на момент постройки/ремонта помещения это не составляет труда, то после введения в эксплуатацию размещение подобных систем вызывает определённые трудности;
- Невозможность пользователю самостоятельно внедрить в существующую систему новые устройства и модули;
- Высокая стоимость подобных решений, так как требует проектирования на этапе внедрения, а также дорогостоящего коммерческого оборудования.

Однако имеются положительные стороны:

- Пользователь не касается аппаратной настройки устройств и модулей, он получает систему “под ключ”;
- Уровень стабильности подобных систем очень высокой.

2) Открытые решения

В сети интернет имеется множество DIY систем умных домов, основанных на open-source технологиях, типа OpenHAB, Node-RED, MajorDomo.

Минусы:

- Отсутствие собственных аппаратных модулей;
- Визуальная оболочка очень часто или не продумана или отсутствует как таковая.

Плюсы:

- Гибкая настройка системы.
- Большое количество программных модулей для связи с коммерческими решениями;
- Постоянная доработка проектов;
- Возможность пользователю самостоятельно изменять конфигурации и обработку событий.

3) Домашние решения

Домашние решения обычно строятся на открытых платформах подобных Arduino. Часто сводятся к разрозненной системе с огромным количеством проводов, выполняющей ограниченное количество действий ввиду отсутствия тщательной проработки и достаточной опытности разработчика. Что в свою очередь приводит к:

1) Сложности в масштабировании и доработке;
2) Сложности интеграции готовых решений;

Однако автоматизированные системы актуальны не только в жилых помещениях, но и административных, и промышленных, где требуется автоматизация доступа в помещениях, автоматизация освещения, контроль над климатом и управление им и пр.

В данной работе описывается программная система, которая позволит автоматизировать управление системой вентилирования помещений центра молодёжного инновационного творчества Гимназии №1 - “Универс” (далее ЦМИТ). А так же автоматизировать ограничение доступа к помещениям с оборудованием, для лиц не имеющих права работать на нём.

На данный момент установленная система вентилирования в ЦМИТ поддерживает только ручное управление с возможностью включения/выключения и плавной регулировкой скорости воздушного потока. Данное решение не позволяет своевременно включать и регулировать вентилирование, что значительно сказывается на уровне комфорта в помещении. Регулирование происходит исключительно по субъективным

признакам и, зачастую, со значительным опозданием, когда дискомфорт становится очевидным.

Доступ к помещениям со специальным оборудованием осуществляется с использованием обычного механического замка с ключом. Это не позволяет организовать гибкое управление доступом большого количества пользователей.

При внедрении автоматизированной системы управления системами жизнеобеспечения планируется:

- Уменьшить время регулирования системой вентиляции, что позволит сохранять климат помещения на комфортном уровне;
- Организовать электронный доступ к помещениям со специальным оборудованием по паролю или электронному ключу;
- Осуществлять контроль и управление доступом к спец-помещениям.

1. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

1.1. Анализ существующих решений.

Разрабатываемая программная система планируется с открытым исходным кодом, поэтому аналоги будут учитываться только из свободно распространяемых приложений.

Современное сообщество открытого программного обеспечения предлагает не очень большой список программ по автоматизации помещений:

openHAB

Открытая система для создания интерактивных пространств[1]. Имеет поддержку огромного количества коммерческих решений (Samsung SmartTV, Tesla, ZWave и т.п.) которые устанавливаются как дополнения.

Для описания логики взаимодействия имеется встроенный язык с собственным синтаксисом[2].

Достоинства:

- Работа с большим количеством готовых коммерческих решений;
- Наличие мобильного приложения;

Недостатком системы является то, что для описания взаимодействия модулей требуется освоение нового языка программирования;

Node-RED

Open-source детище IBM и, как написано на сайте, это инструмент, который служит для связи железа, API и сервисов новыми и интересными способами[3].

Способы эти основаны на использовании графических линий связи, по которым пересылаются сообщения между узлами. Таким образом связывать различные блоки можно просто мышкой без использования программирования.

Разработка в Node-RED ведется через обыкновенный браузер, само ядро можно запустить на различных платформах – PC, RPi, cloud и т.д.

Разрабатываемая программа рассчитана на старших школьников, которые только на начальном уровне изучают программирование. Т.е. ПО должно быть

интуитивно-понятно, просто в обращении и в случае, когда его возможностей не хватает была возможность добавить (расширить) функционал.

Итак, в данном подразделе были найдены и проанализированы альтернативные программные продукты, принято решение о разработке программной системы управления системами жизнеобеспечения помещений. Основными преимуществами разрабатываемой системы является то, что она предоставляет возможность лёгкого масштабирования. ПО использует простейшие сценарии, которые не требуют изучение дополнительных средств разработки. Так же плюсом является то, что система имеет собственную библиотеку для модулей ESP8266, которые сравнимы по стоимости с платами Arduino, но имеют более высокую производительность, а также встроенный модуль WiFi.

Далее, для успешной реализации проекта, необходимо осуществить анализ и выбрать средство разработки.

1.2. Выбор средств разработки

Выбор средств разработки осуществляется на первых стадиях работы над проектом, после определения требований к создаваемому программному продукту. Он, как и анализ существующих альтернативных программных продуктов, является экономическим процессом, в ходе которого выбираются наиболее оптимальные средства разработки для конкретного проекта. Например, по соотношению стоимости средства к необходимым функциям, по времени и стоимости обучения работы со средством разработки и т.д.

Основной составляющей средств разработки является инструментальное программное обеспечение (Software tools) – программное обеспечение, используемое в ходе разработки, корректировки или развития других программ. Сюда входят языки программирования, интегрированные среды разработки, а также различные вспомогательные программы.

Существующие на сегодняшний день языки программирования можно выделить в следующие группы:

- универсальные языки высокого уровня;
- специализированные языки разработчика ПО;
- специализированные языки пользователя;
- языки низкого уровня.

Для разработки программы системы управления необходим универсальный язык программирования, позволяющий производить связь различных модулей и программировать их взаимодействие. Подобными языками являются: C++, Visual Basic, Java, C#, Object Pascal и другие.

C++

Является лидером по количеству использования в проектах на сегодняшний день.

Его основные достоинства это:

- кроссплатформенность, т.е. возможность разрабатывать программы для самых разных платформ и систем;
- возможность работать на низком уровне с памятью, адресами, портами, что при неосторожном использовании может легко превратиться в недостаток;
- мощный препроцессор, унаследованный от языка C, но, как и любой другой мощный инструмент, требует осторожного обращения;
- возможность создания обобщенных алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции и другие достоинства.

При этом язык C++ обладает рядом недостатков:

- подключение интерфейса внешнего модуля через препроцессорную ставку заголовочного файла (#include) серьезно замедляет компиляцию при подключении большого количества модулей;
- недостаток информации о типах данных во время компиляции;
- сложность для изучения и компиляции;
- некоторые преобразования типов не интуитивны. В частности, операция над беззнаковыми и знаковыми числами дает беззнаковый результат.

Visual Basic

Часто используется для второстепенных проектов, позволяющий с высокой скоростью создавать приложения с графическим интерфейсом для операционных систем семейства Microsoft Windows, имеющий простой для изучения синтаксис. Также имеется возможность компиляции программы, как в машинный код, так и в P-код (по выбору программиста). В режиме отладки программа всегда (вне зависимости от выбора) компилируется в P-код, что позволяет приостанавливать выполнение программы, вносить значительные изменения в исходный код, а затем продолжать выполнение: полная перекомпиляция и перезапуск программы при этом не требуется. P-код – это концепция аппаратно-независимого исполняемого кода, который также можно назвать байт-код. Недостатками языка являются:

- поддержка операционных систем только семейства Windows и Mac OS X (Исключение — VB1 for DOS);
- отсутствие механизма наследования реализации объектов. Существующее в языке наследование позволяет наследовать только интерфейсы, но не их реализацию;
- медленная скорость работы, обусловленная тем, что практически все встроенные функции языка реализованы через библиотеку времени исполнения (runtime library), которая, в свою очередь, производит много «лишней» работы по проверке и/или преобразованию типов;
- возможность отключить средства слежения за объявленными переменными, возможность неявного преобразования переменных, наличие типа данных «Variant». По мнению критиков, это даёт возможность писать крайне плохой код;
- отсутствие указателей, низкоуровневого доступа к памяти, ассемблерных вставок. Несмотря на то, что парадигма Visual Basic позволяет среднему VB-программисту обходиться без всего этого, перечисленные вещи также нередко становятся объектами критики.

Java

Объектно-ориентированный язык программирования, разработанный компанией «Sun Microsystem» в 1995 году. Скомпилированные в байт-код программы Java работают только под управлением виртуальной Java-машины (JVM), поэтому они называются приложениями Java. JVM — программа, обрабатывающая байтовый код и передающая инструкции оборудованию как интерпретатор, но с тем отличием, что байтовый код, в отличие от текста, обрабатывается значительно быстрее. Достоинство подобного способа выполнения программ — в полной независимости байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание. Язык гораздо сильнее типизирован по сравнению с C++, т.е. вносит больше ограничений на действия с переменными и величинами различных типов. Наличие готовых шаблонов проектирования (patterns), которые представляют собой многократно используемые решения широко распространенных проблем, позволяют повышать производительность работы программистов. Также одним из достоинств является наличие большого количества профессиональных и бесплатных сред разработки, поддерживающих данный язык.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. Данное утверждение было справедливо для первых версий виртуальной машины Java, однако в последнее время оно практически потеряло актуальность, благодаря значительному количеству качественных усовершенствований.

Из-за отсутствия хорошего и законченного средства создания форм язык пока остается малопригоден для разработки оконных приложений. В настоящий момент основная область его применения – сетевые технологии: создание клиент-серверных приложений.

C#

Также объектно-ориентированный язык программирования. Разработан корпорацией Microsoft в 1998-2001 годах. C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, направленное на возможность повторного использования созданных компонентов. Язык является проще и надежнее C++, реализует возможности наследования и универсализации и тесно связан с мощной средой Framework .Net. Благодаря каркасу Framework. Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда для C# представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода.

Недостатками C# являются влияние на использование памяти и производительность. Динамическая компиляция с проверкой во время выполнения работает медленнее, чем код, написанный на C/C++. Периодически должны выполняться процедуры сборки мусора для восстановления памяти, и возможно даже удаления некоторого кода из кэша кода. Во время сборки мусора все потоки должны приостанавливаться, и это может иметь отрицательное влияние на производительность системы в реальном времени.

Delphi (Object Pascal)

Язык разработан в фирме Apple Computer в 1986 году на базе языка Pascal. Позже был значительно доработан фирмой Borland и в официальной документации компании стал упоминаться, как язык Delphi. Является объектно-ориентированным языком. Его основные достоинства:

- позволяет писать лаконичный и безопасный (за счет строгой типизации) код;
- компилятор языка обнаруживает большое количество не только синтаксических, но и семантических ошибок;
- реализована возможность использования ассемблерных вставок в коде, что позволяет существенно сократить время выполнения определенных функций в создаваемой программе;
- существует возможность использовать библиотеку визуальных компонентов (VCL). VCL – объектно-ориентированная библиотека для разработки программного обеспечения, разработанная компанией Borland для поддержки принципов визуального программирования. Она предоставляет огромное количество готовых к использованию компонентов для работы в самых разных областях программирования.

Недостатками языка являются:

- необходимость в некоторых ситуациях использовать довольно громоздкие конструкции языка, что приводит к снижению читаемости кода. Подробное комментирование кода позволяет компенсировать данный недостаток;
- слабая поддержка методов работы с оперативной памятью и другими ресурсами компьютера;
- ограниченное количество сред разработки, поддерживающих данный язык (а фактически – две);

Проанализировав достоинства и недостатки языков, осуществим выбор языка программирования, с помощью которого будем создавать ПО. В большей степени выбор языка программирования определяется опытом и знаниями конкретного языка разработчиком, предназначением и функциональными особенностями разрабатываемого ПО. С учетом достаточных знаний и опытом практического применения языка C++ и фреймворка Qt на различных проектах, было принято решение использовать именно его для разработки программы в рамках дипломного проектирования. Возможности данного языка и Qt, в виду

наличия большого количества библиотек по обмену данными, делают их сочетание очень гибким инструментом для создания системы управления различными аппаратными модулями. Система сигналов и слотов Qt позволяет гибко настраивать выполнение команд при наличии входной информации.

К тому же, средство Qt Creator и фреймворк Qt являются открытыми и бесплатными для использования в проектах с открытой лицензией. Ещё одним плюсом будет кроссплатформенность данного фреймворка, что позволит с большей лёгкостью переносить подобные системы на различные платформы.

1.3. Постановка задачи

Реализовать программную систему состоящую из программного обеспечения, которое позволит настраивать и управлять взаимодействием между объектами системы и библиотеку для контроллеров Espressif ESP8266, которая позволит использование в составе программной системы.

Разрабатываемая программная система должна отвечать следующим требованиям:

1) Кроссплатформенность. Возможность работы на операционных семейства Windows не младше 7 версии, *nix операционных системах и OS X;
2) Возможность настройки поведения каждого объекта, а также их взаимодействие;
3) Выполнение команд по расписанию и таймеру;
4) Вызов внешних процессов с параметрами;
5) Графический пользовательский интерфейс;
6) Автоматический поиск модулей в локальной сети;
7) Наличие справочного материала по работе с программной системой;

1.4. Выбор протокола передачи данных

Сегодня стек TCP/IP используется повсеместно как в глобальных, так и в локальных сетях. Этот стек имеет иерархическую структуру, в которой определено 4 уровня представленных в таблице 1[4].

Таблица 1 – Стек протоколов TCP/IP

Уровень	**Протоколы**
Прикладной уровень	FTP, Telnet, HTTP, SMTP, SNMP, TFTP
Транспортный уровень	TCP, UDP
Сетевой уровень	IP, ICMP, RIP, OSPF
Уровень сетевых интерфейсов	Не регламентируется

Прикладной уровень стека TCP/IP соответствует трем верхним уровням модели OSI: прикладному, представления и сеансовому. Он объединяет сервисы, предоставляемые системой пользовательским приложениям.

Транспортный уровень стека TCP/IP может предоставлять вышележащему уровню два типа сервиса:

- гарантированную доставку обеспечивает протокол управления передачей (Transmission Control Protocol, TCP);
- доставку по возможности, или с максимальными усилиями, обеспечивает протокол пользовательских дейтаграмм (User Datagram Protocol, UDP).

Сетевой уровень, называемый также уровнем Интернета, является стержнем всей архитектуры TCP/IP. Именно этот уровень, функции которого соответствуют сетевому уровню модели OSI, обеспечивает перемещение пакетов в пределах составной сети, образованной объединением нескольких подсетей. Протоколы сетевого уровня поддерживают интерфейс с вышележащим транспортным уровнем, получая от него запросы на передачу данных по составной сети, а также с нижележащим уровнем сетевых интерфейсов.

Сетевой уровень, называемый также уровнем Интернета, является стержнем всей архитектуры TCP/IP. Именно этот уровень, функции которого соответствуют сетевому уровню модели OSI, обеспечивает перемещение пакетов в пределах составной сети, образованной объединением нескольких подсетей.

Уровень сетевых интерфейсов отвечает только за организацию взаимодействия с подсетями разных технологий, входящими в составную сеть. TCP/IP рассматривает любую подсеть, входящую в составную сеть, как средство транспортировки пакетов между двумя соседними маршрутизаторами.

2. РАЗРАБОТКА СИСТЕМЫ

2.1. Архитектура программной системы

Принцип взаимодействия объектов ПО GeekSpace заимствован из фреймворка Qt, где для связи объектов используются т.н. сигналы и слоты. Сигнал вырабатывается, когда происходит определенное событие. Слот - это функция, которая вызывается в ответ на определенный сигнал, но при этом он может использоваться как обычная функция[5]. Данная реализация позволяет гибко и безопасно настраивать взаимодействие между объектами системы путём соединения сигналов со слотами. Для соединения сигнала со слотом необходимо вызвать контекстное меню нужного сигнала/слота и в появившемся меню выбрать необходимый слот/сигнал.

Каждый из объектов должен иметь своё уникальное имя в системе, для того, чтобы ПО могло правильно идентифицировать каждый из объектов и в случае необходимости обратится к нему. Уникальность создания имён достигается путём передачи имён существующих объектов в диалоговые окна добавления объектов, а диалоговое окно в свою очередь блокирует создание объекта с уже существующим именем.

Для удобства хранения и стандартизации вызова слотов и получения сигналов объекты наследуются от общего абстрактного класса GSObject, в котором реализованы общие функции для всех типов объектов.

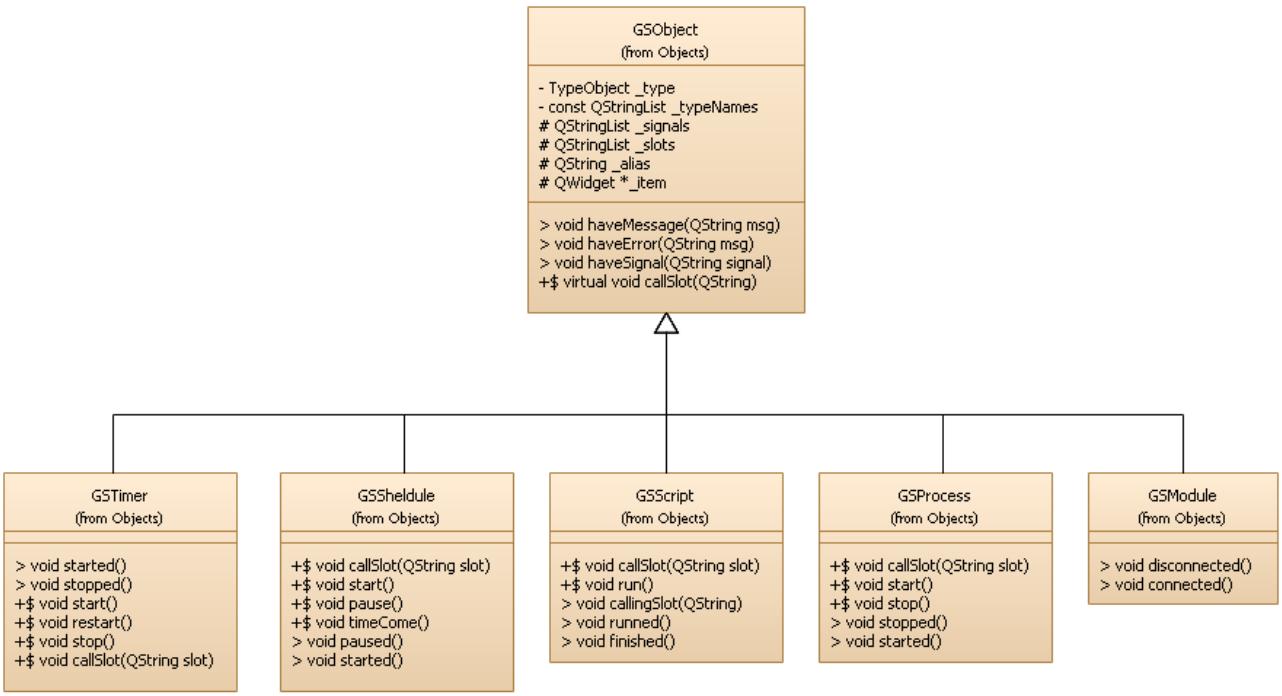

Рисунок 2.1.1 – Диаграмма наследования классов объектов системы GeekSpace.

На рисунке 2.1.1 изображена диаграмма наследования объектов. Из неё можно увидеть, что родительский класс GSObject содержит следующие сигналы:

- haveMessage(QString msg) - наличие информационного сообщения от объекта;
- haveError(QString msg) - наличие ошибки в работе объекта, которое требует внимания пользователя;
- haveSignal(QString signal) - наличие сигнала объекта для обработки системой.

Общая структура взаимодействия классов системы представлена на следующей диаграмме:

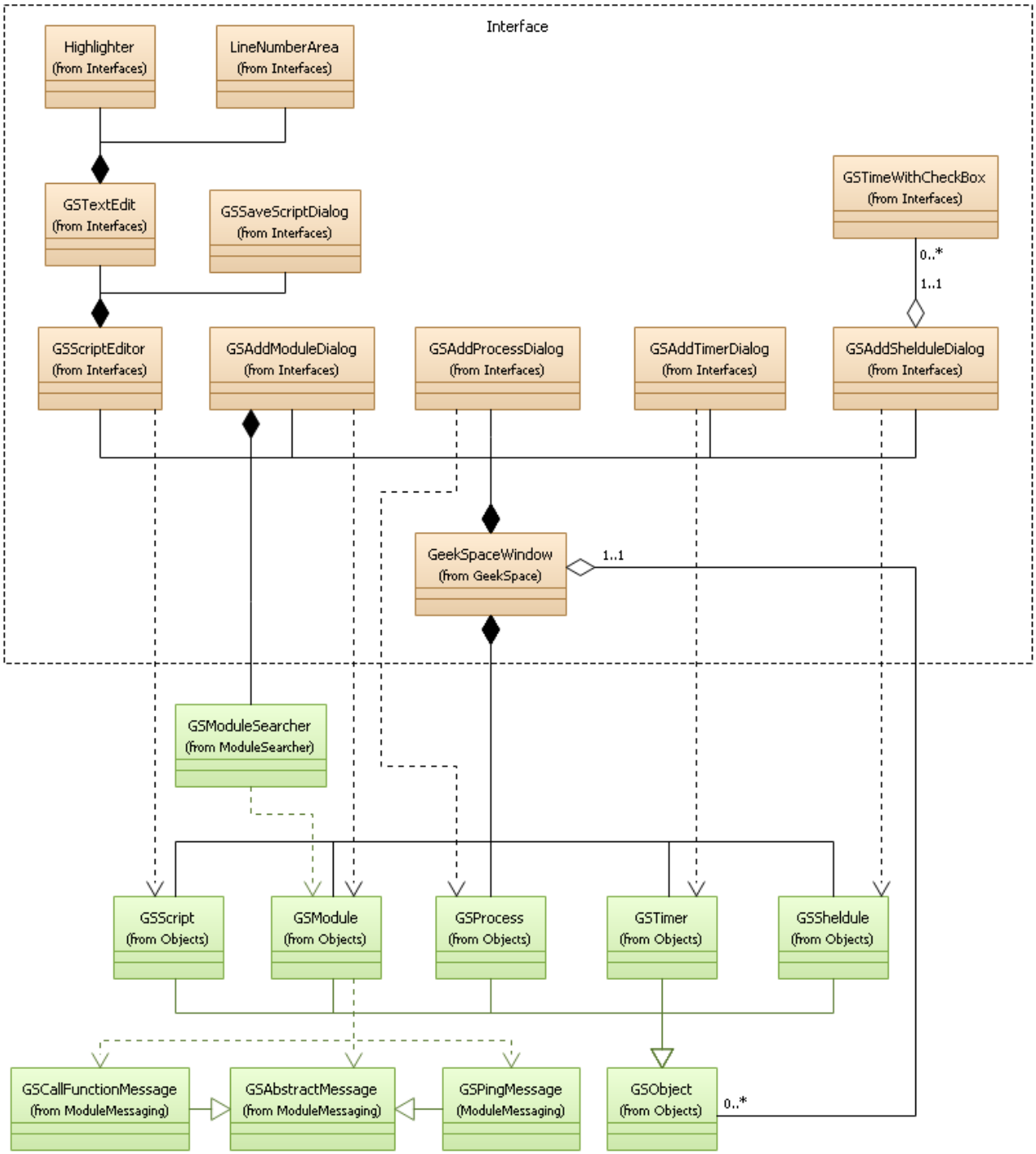

Рисунок 2.1.2 – Диаграмма взаимодействия классов системы GeekSpace.

2.2. Интерфейс

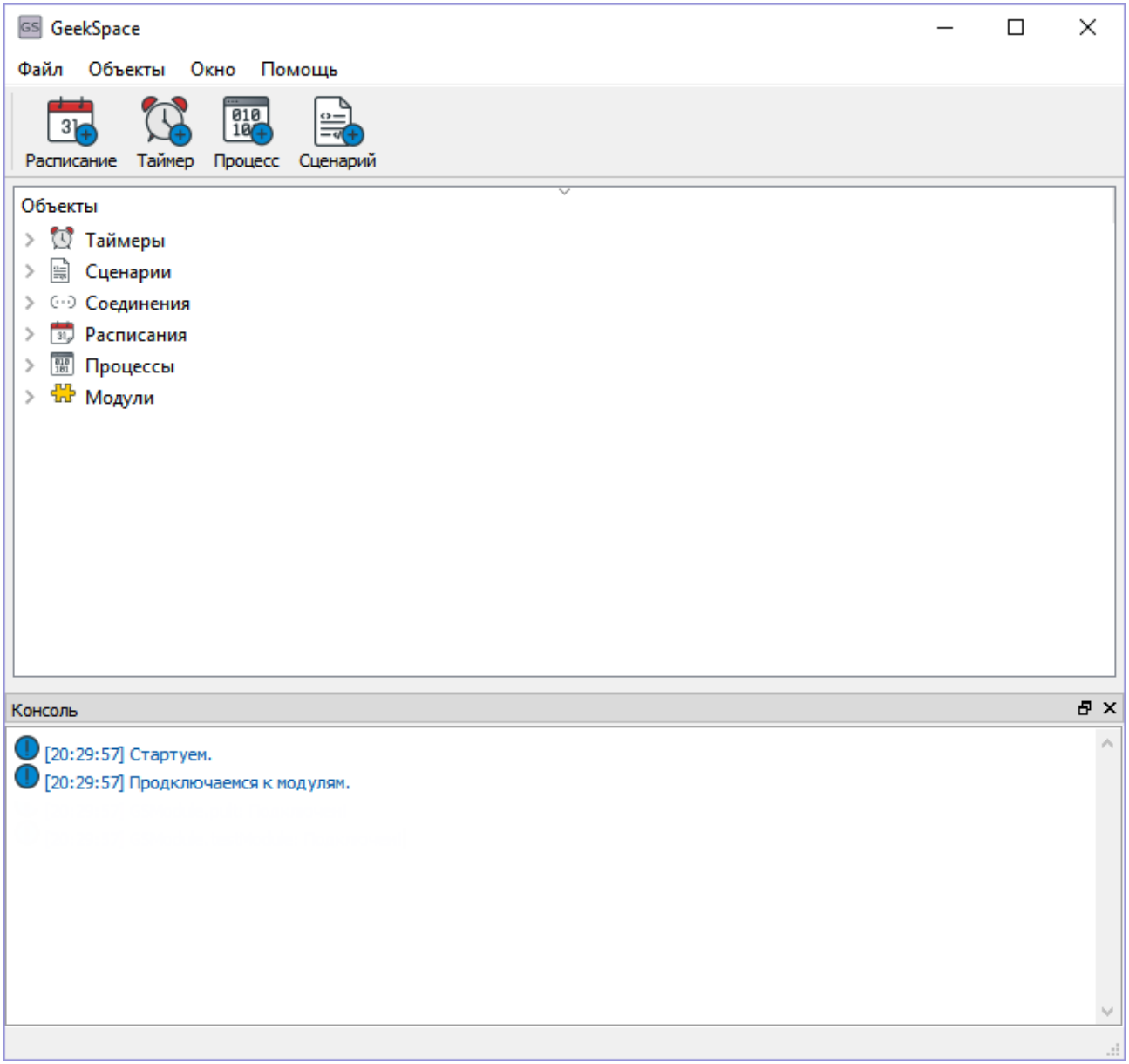

Рисунок 2.2.1 – Главное окно графического интерфейса ПО GeekSpace

Главное окно графического пользовательского интерфейса ПО GeekSpace разделено на две области:

- Древовидный виджет, в котором выводятся все объекты системы, сгруппированные по типу.
- Консоль - текстовое поле для вывода сообщений системы, таких как запуск или остановка процессов, потеря соединения с модулем и т.п.

Окно имеет главное меню, в котором доступны все действия программы. Так же в окне присутствует панель инструментов, на которой расположены кнопки добавления объектов.

2.3. Реализованный функционал

В программном обеспечении описаны алгоритмы, реализующие возможности:

- Добавление и редактирование объектов;
- Установление и редактирование взаимосвязей между объектами через контекстное меню;
- Автоматическое сохранение при выходе и загрузка при старте системы всех объектов системы;

Для реализации требований указанных выше были реализованы следующие объекты:

2.3.1. Расписание

Объект расписание создан для генерации сигналов привязанных к определённым датам или дням недели и времени. Имеет следующие возможности:

- Вызов сигнала единожды в заданную дату и время
- Вызов сигнала через заданное время:
    - День;
    - Неделя;
    - Месяц;
    - Год.
- Настройка расписания по времени на неделю. Для каждого дня недели можно создать неограниченное количество срабатываний по времени.

Сигналы расписания:

- timeCome() - свидетельствует о наступления запланированного времени.

Слоты:

- start() - запуск расписания;
- pause() - остановка расписания;

Для добавления объекта расписание реализован диалог с двумя вкладками. Первая отвечает за выполнение расписания в назначенную дату и время.

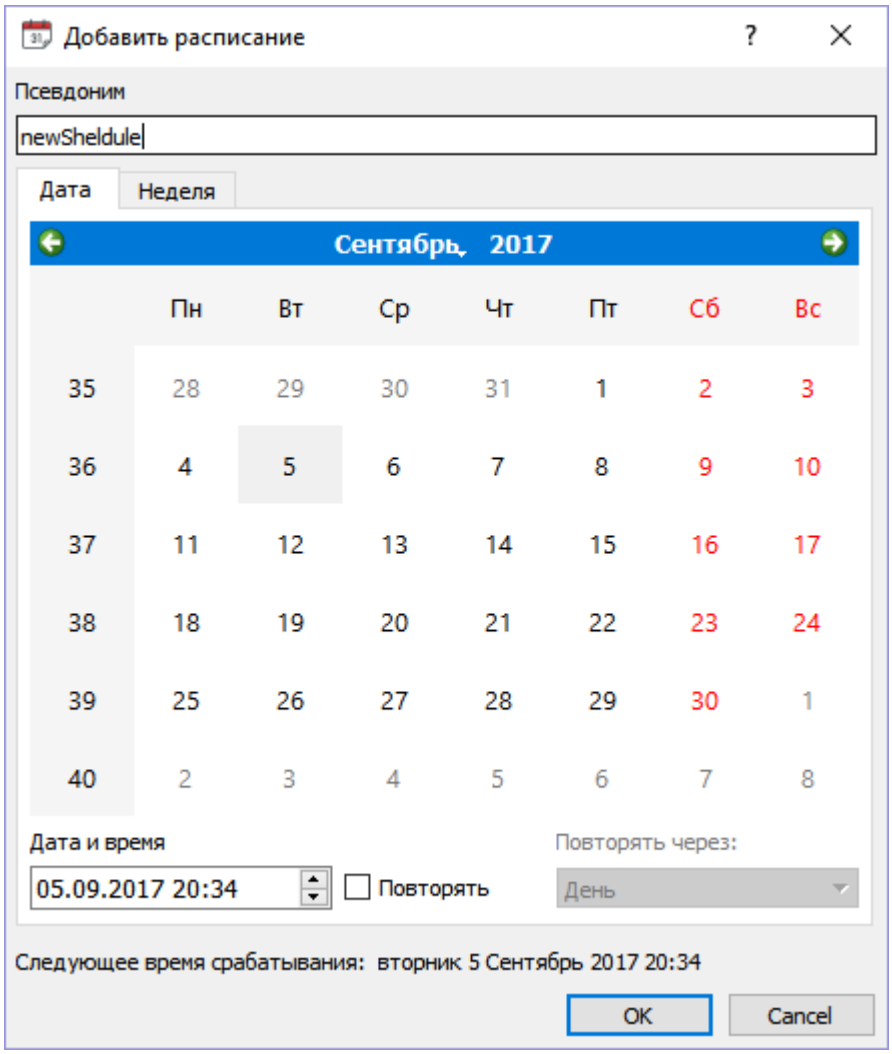

Рисунок 2.3.1.1 – Диалог добавления объекта расписания, вкладка “Дата”

Вторая вкладка позволяет устанавливать расписание на неделю. Добавление столбцов времени происходит автоматически, при установке элемента “флажок” в отмеченное состояние.

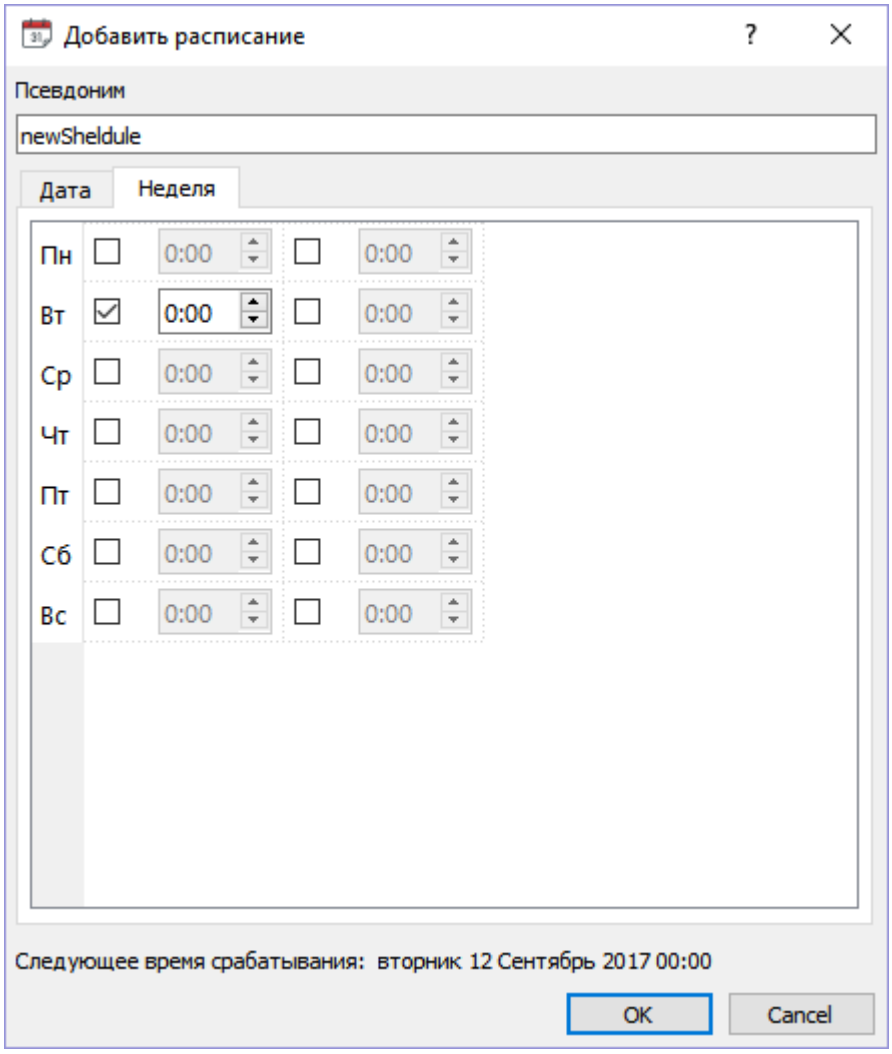

Рисунок 2.3.1.2 – Диалог добавления объекта расписания, вкладка “Неделя”

2.3.2. Таймер

Таймер позволяет генерировать сигнал через указанный промежуток времени и имеет возможность перезапуска бесконечно или заданное количество раз;

Сигналы расписания:

- timeout() - свидетельствует об окончании времени.

Слоты:

- start() - запуск таймера;
- stop() - остановка таймера;

- restart() - перезапуск таймера со сбросом счётчика количества выполнения;

Объект таймер добавляется через диалог “Добавить таймер”.

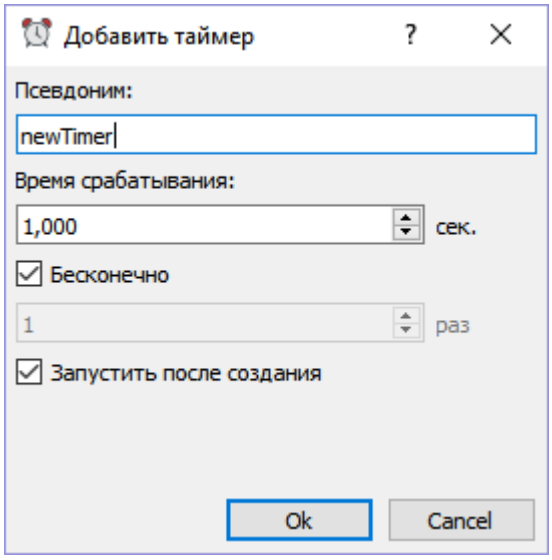

Рисунок 2.3.2.1 – Диалог добавления объекта таймер

Диалог добавления таймера позволяет установить период срабатывания с точностью до одной миллисекунды. А так же количество повторений: бесконечно или заданное количество раз.

2.3.3. Процесс

Объект процесс создан для расширения функционала программной системы и запуска внешних приложений. В связке с такими программами как nircmd[6] и Auto Mouse Clicker[7] можно получить полный контроль над системой через модули.

Сигналы процесса:

- started() - вызывается, когда приложение запущено;
- stopped() - при завершении работы процесса.

Слоты:

- start() - запуск программы;
- stop() - остановка программы;

Диалог добавления таймера имеет следующий вид:

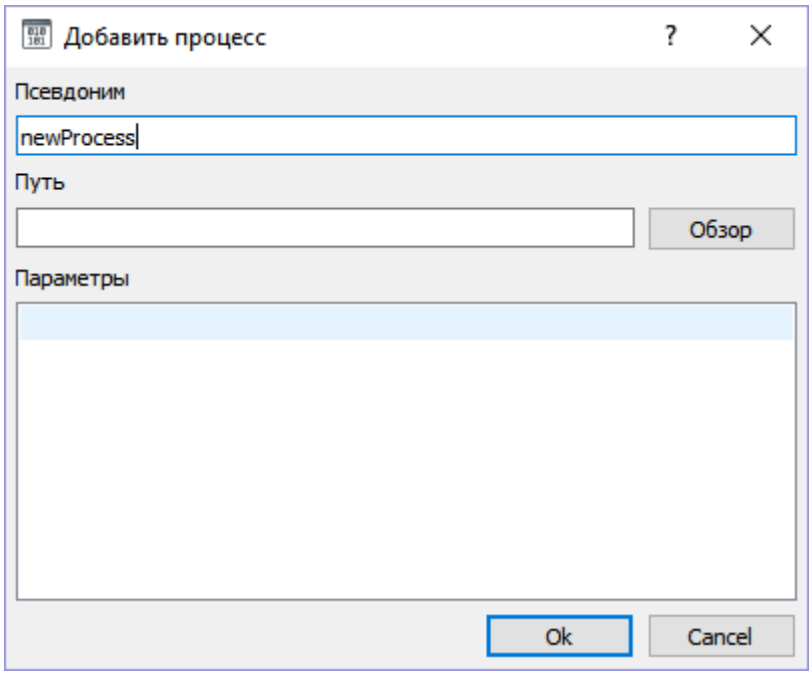

Рисунок 2.3.3.1 – Диалог добавления объекта таймер

2.3.4. Сценарий

Сценарий как объект позволяет скомпоновать несколько выполняемых слотов в один, что позволяет экономить время при создании типовой задачи. Для редактирования сценария разработан редактор сценариев:

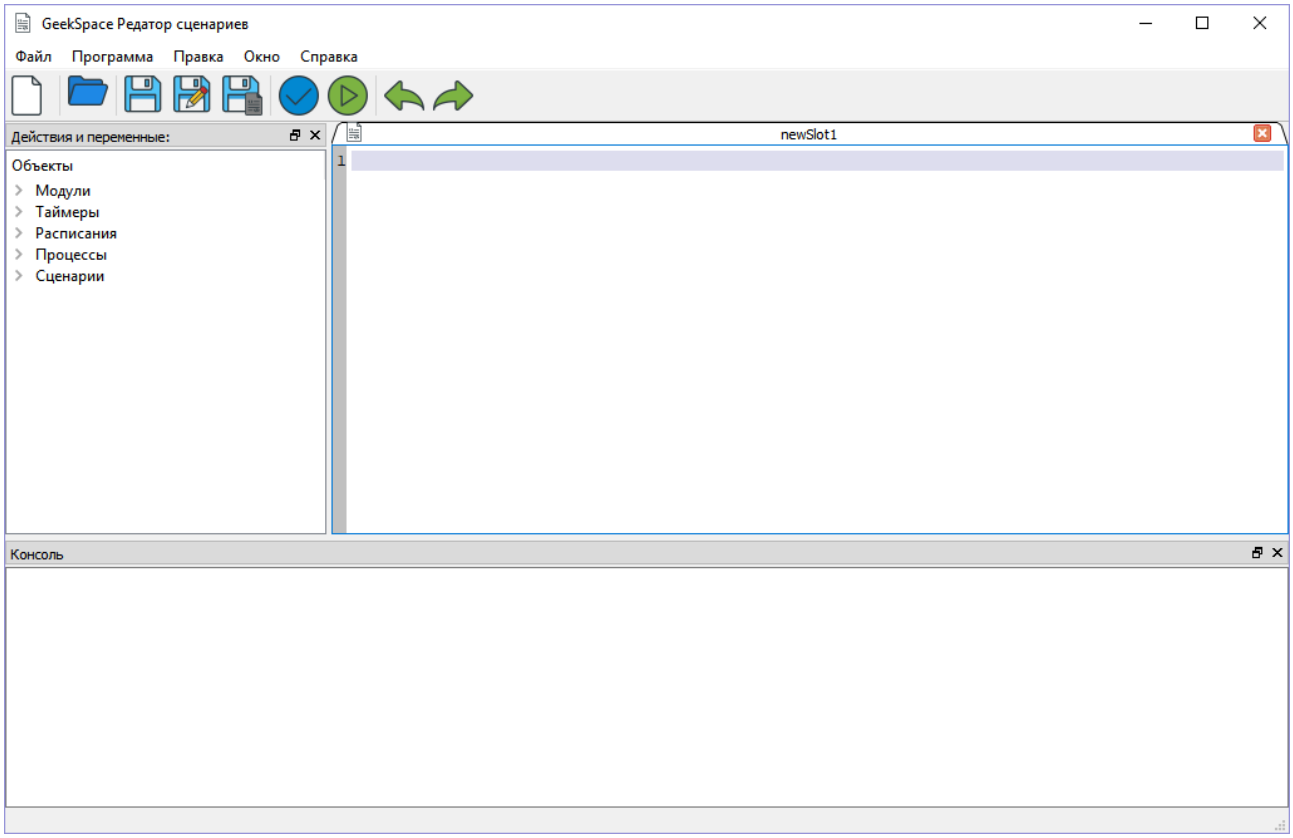

Рисунок 2.3.4.1 – Окно редактора сценариев

Редактор имеет возможность изменения сразу нескольких сценариев. Левая панель содержит все объекты системы и их слоты. По двойному клику по слоту в текст сценария добавляется строка с вызовом данного слота, что позволяет быстро создавать текст сценария.

Для того чтобы проверить на правильность подготовленного сценария можно запустить проверку Главное меню -> Программа -> Проверить или выполнить тестовый запуск Главное меню -> Программа -> Выполнить для того чтобы оценить правильность выполнения сценария. Если Ошибок в сценарии не имеется, то в консоль редактора будет выведено сообщение “Проверка закончилась удачно.”. В противном случае редактор сообщит о наличии ошибки написав ст

2.3.5. Модуль

Объект модуля один из самых важных в разрабатываемой системе, так как именно он обеспечивает управление системами жизнеобеспечения. Однако

по умолчанию в системе он не имеет встроенных сигналов и слотов, а пользователь самостоятельно их создаёт по необходимости. Процесс разработки описан в разделе 2.7.5 Руководства программиста.

2.4. Протокол связи с модулем

Система сообщений GeekSpace для обмена данными с модулем содержит данные:

- Размер сообщения - тридцати двух битное беззнаковое число;
- Код типа сообщения - восьми битное число;
- Дополнительные данные - различные данные различной длинны

GeekSpace использует следующие коды типа сообщений:

- 0x00 - пустое сообщение (не используется);
- 0x01 - сообщение Ping;
- 0x02 - сообщение Pong (ответ на Ping);
- 0x03 - вызов слота модуля, добавляется код вызываемого слота;
- 0x04 - не используется;
- 0x05 - информация, добавляется текстовая информация в формате XML содержащая всю информацию о модуле;
- 0x06 - вызов сигнала, дополнительно указывается код сигнала.

2.6. Библиотека GeekSpace Module

Библиотека GeekSpace Module представляет собой набор классов-оболочек над основными классами коммуникации модуля ESP8266. Позволяет с минимальным количеством кода осуществить подключение модуля к ПО GeekSpace. Классы библиотеки имеют следующую структуру:

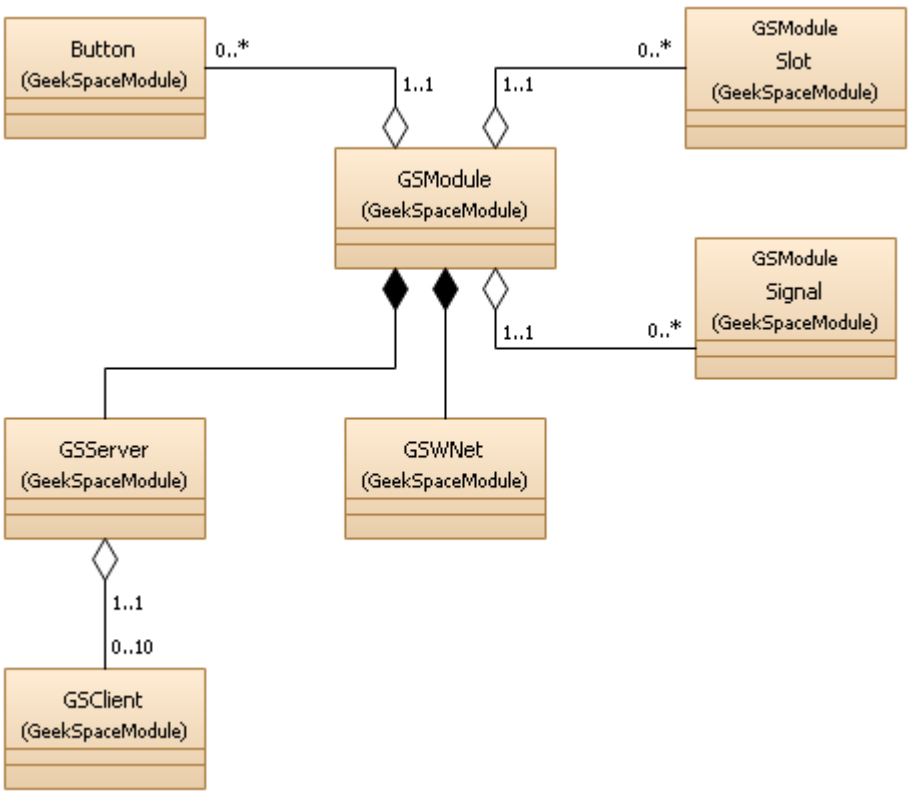

Рисунок 2.6.5 – Диаграмма взаимодействия классов библиотек GeekSpace Module.

Библиотека состоит из следующих объектов:

- GSModule - предоставляет набор функций для взаимодействия разработчика модуля и ПО GeekSpace;
    - Button - отслеживает состояния пина кнопки и предоставляет доступ к этим состояниям;
    - Slot - структура, в которой хранится информация о добавленном слоте: название, код, ссылка на функцию;
    - Signal - структура для хранения информации о добавленном сигнале.
- GSWNet - класс-обёртка над библиотекой ESP8266WiFi, отвечает за подключения модуля к точке доступа по заданному идентификатору сети и паролю;
- GSServer - класс-обёртка над классом WiFiServer для отслеживания входящих подключений ПО GeekSpace. Также прослушивает UDP порт для обнаружения модуля в локальной сети для ПО GeekSpace;

- GSClient - класс-обёртка над классом WiFiClient. Отвечает за передачу данных между модулем и ПО GeekSpace.

2.7. Руководство программиста

2.7.1. Общие сведения о программе

Программная система GeekSpace предназначенная для предоставления пользователю возможности создания автоматизированной системы управления системами жизнеобеспечения помещения с возможностью её контроля и настройки.

Данная система состоит из двух частей:

1) ПО GeekSpace - программа для управления и описания взаимодействия между объектами системы;
2) GeekSpace Module - библиотека для аппаратных модулей на основе чипов Espressif ESP8266 для подключения модулей к ПО GeekSpace.

2.7.2. Характеристика системы

ПО GeekSpace разработано как интерфейсная программа. Позволяет пользователю через графический интерфейс настраивать и описывать взаимодействия различных аппаратных модулей, а также других объектов системы.

2.7.3. Настройка системы

Настройка системы производится путём добавления в ПО GeekSpace различных объектов, а также разработкой кода для аппаратных модулей на базе контроллера ESP-8266. Работа с объектами системы описана в разделе 2.8 Руководство пользователя.

2.7.4. Сообщения

Сообщения ПО GeekSpace выводятся в консоль главного экрана приложения и имеют формат: Иконка_типа_сообщения [Время_сообщения] Текст_сообщения.

Рисунок 2.7.4.1 – Иконки типов сообщений.

Сообщения можно разделить на два типа:

1. Информация - сообщения, которые несут информацию об изменении в состоянии системы или её объектах отмечаются иконкой как показано на рисунке Рисунок 2.7.1 слева и имеют синий цвет текста;
2. Ошибки - данные сообщения говорят о неправильной работе системы, которые могут привести к нежелательным или неожиданным последствиям работы системы. Обозначаются, как показано на рисунке Рисунок 2.7.1 справа и имеют красный цвет текста.

ПО GeekSpace содержит следующие информационные сообщения:

- "Стартуем" - Запуск системы, загрузка объектов;
- "Подключаемся к модулям" - Начат процесс подключения к модулям;
- "GSModule.%1: Подключен!" - Произведено подключение к модулю %1, где %1 это псевдоним модуля;
- "Добавлено модулей: %1" - Произведено добавление модулей, где %1 это количество модулей;
- "GSProcess.%1: Процесс \"%2\" запущен" - Сообщение об успешном запуске приложения, где %1 псевдоним процесса, а %2 путь до него;
- "GSProcess.%1: Процесс \"%2\" остановлен" - Сообщение о завершении выполнения процесса, где %1 псевдоним процесса, а %2 путь до него;
- "GSSheldule.%1: Время пришло." - Сообщение объекта расписание о том, что наступили выбранные дата и время, где %1 псевдоним расписания.
- "GSSheldule.%1: Следующее время срабатывания %2." - Сообщение о том, когда будет произведено следующие срабатывание сигнала timeCome, где %1 псевдоним расписания, а %2 следующие время срабатывания.

Информационные сообщения редактора сценариев:

- "Проверка закончилась удачно." - Произведена проверка текста сценария и ошибок не найдено.
- "Файл записан в '%1'." - Запрос на сохранение файлы был выполнен удачно и файл записан по адресу %1;

  Сообщения об ошибках ПО GeekSpace:
- "GSObject.%1: Неправильный вызов слота" - Внутренняя ошибка программы, требуется сообщить разработчику;
- "GSModule.%1: Вызов неизвестного слота \"%2\"." - Системой была предпринята попытка вызова слота модуля, который был модифицирован или удалён. Требуется пересоздать соединение или удалить его. %1 - псевдоним модуля, %2 - имя вызываемого слота.
- "GSModule.%1: Получен неизвестный сигнал: 0x%2" - От модуля был получен сигнал, которого нет в системе. Данное сообщение может появляется, если система не обновила данные о модуле, а внутренняя программа модуля была изменена. Или имеется ошибка в описании внутренней программы модуля. Рекомендуется перезапустить систему. Если перезапуск не исправляет ошибку, то проверить исходный код внутренней программы модуля. %1 - псевдоним модуля, %2 код сообщения в шестнадцатеричном формате;
- "GSModule.%1: Удалённый хост закрыл соединение." - Ошибка связи с модулем. Модуль был отключен или была потеряна с ним связь. Необходимо проверить питание модуля и доступность сети WiFi в зоне размещения модуля. %1 - псевдоним модуля;
- "GSModule.%1: Ошибка подключения: Хост не найден." - Ошибка подключения или переподключения к модулю. У модуля изменился адрес. Проверить доступность модуля по текущему адресу или см причины и способы их устранения выше. %1 - псевдоним модуля;
- "GSModule.%1: Соединение разорвано модулем." - Ошибка в библиотеке модуля. Требуется сообщить об ошибке разработчику. %1 - псевдоним модуля;

- "GSModule.%1: Ошибка сокета: %2." - Возникает при прочих ошибках связи с модулем. Решение зависит от обстоятельств и текста ошибки. %1 - псевдоним модуля, %2 текст ошибки сокета
- "GSProcess.%1: Ошибка запуска процесса \"%2\"" - Появляется при невозможности запустить процесс. Необходимо проверить путь до выполняемой программы. %1 псевдоним процесса, %2 путь до процесса;
- "GSProcess.%1: Процесс \"%2\" не запущен" - Ошибка остановки процесса при попытке остановить не запущенный процесс. %1 псевдоним процесса, %2 путь до процесса;

Ошибки редактора сценариев:

- "Ошибка чтения файла сценария '%1' при редактировании." - Ошибка возникает когда невозможно загрузить текст сценария, который хранится в папке пользователя в подкаталоге “\GeekSpace\Scripts”. Это возможно при повреждении файловой системы или при удалении файла текста скрипта. Необходимо проверить диск на ошибки или попытаться восстановить удалённый файл. %1 - имя файла сценария;
- "Проверка закончилась с ошибками." - Сообщение появляется при наличии ошибки в тексте сценария. Необходимо исправить ошибку в тексте программы. Более подробная информация о найденных ошибках выводится в предшествующих данному сообщениях;
- "Ошибка: в строке \"%1\". Синтаксическая ошибка." - Сообщает о наличии в тексте сценария синтаксической ошибки. %1 - текст команды содержащий ошибку;
- "Ошибка: в строке \"%1\". Объект \"%2\" не имеет слота \"%3\"" - Свидетельствует о том, что у используемого в сценарии объекта не имеется указанного слота. %1 - текст команды с ошибкой, %2 - псевдоним объекта, %3;
- "Ошибка: в строке \"%1\". Объект \"%2\" не найден" - Свидетельствует об использовании в сценарии не существующего объекта. %1 - текст команды с ошибкой, %2 - псевдоним объекта;

- "Ошибка записи файла '%1'." - Ошибка возникает при невозможности записи файла на диск вследствии запрета на запись в папку пользователя в подкаталог “\GeekSpace\Scripts”. Или недостаточного места на диске. %1 имя записываемого файла.
- "Невозможно выполнить скрипт \"%1\"." - возникает при наличии ошибок в тексте сценария. Необходимо исправить ошибки.

2.7.5. Разработка модуля

Модуль реализуется на базе контроллера Espressif ESP8266 через среду Arduino IDE. Для использования модуля в составе программной системы GeekSpace необходимо подключить заголовочный файл "gsmodule.h", в котором описаны основные функции необходимые для добавления слотов и сигналов для модуля.

Далее описывается переменная класса, входным параметром которой является псевдоним модуля.

Например:

```
GSModule module("testModule");
```

Для добавления слота используется функция “addSlot”, а параметрами передаются имя слота, которое будет отображено в системе, а также ссылка на функцию, которая будет выполняться при вызове слота. Пример использования:

```
module.addSlot("ledOn", ledOn);
```

Сигнал добавляется через функцию класса “addSignal” и в качестве параметра принимает имя сигнала. Используется следующим образом:

```
module.addSignal("buttonClicked");
```

Чтобы обеспечить пользователя удобными функциями обработки нажатия кнопок, был разработан класс Button, который может проверять нажатие кнопки подключенной к соответствующему типу. Перед определение новой кнопки требуется создать глобальную переменную класса Button, а затем привязать нужный пин. Сделать это можно следующим образом:

```
Button *bOn; //Создаём глобальную переменную класса;
bOn = module.addButton(16); //Создаём объект класса Button для контроля за пином 16.
```

Теперь для проверки состояния кнопки необходимо вызвать нужную функцию, например в условии:

```
if(bOn->isClicked()) { // Проверяем нажатие кнопки
    module.emitSignal("buttonClicked"); // Если нажатие было, то сообщаем системе о нажатии
}
```

2.8. Руководство пользователя

GeekSpace

GeekSpace это программное обеспечение, которое позволяет организовать взаимодействие между компьютером и системой жизнеобеспечения помещения. Под такими системами понимаются:

- Водоснабжение;
- Электроснабжение;
- Климат (кондиционирование и вентиляция);
- Контроль доступа;
- Противопожарная сигнализация;
- Охранная сигнализация;

Данное ПО предлагает настройку взаимодействия между аппаратными модулями, которые реализованы на базе контроллера ESP8266, и компьютером, по средствам сигналов и слотов.

Сигналы это псевдофункции, которые вызываются при наступлении определённых событий объектов системы. Слоты это функции выполняющие определённые действия. При вызове сигнала программа находит все слоты соединённые с ним и вызывает их всех по порядку.

Интерфейс

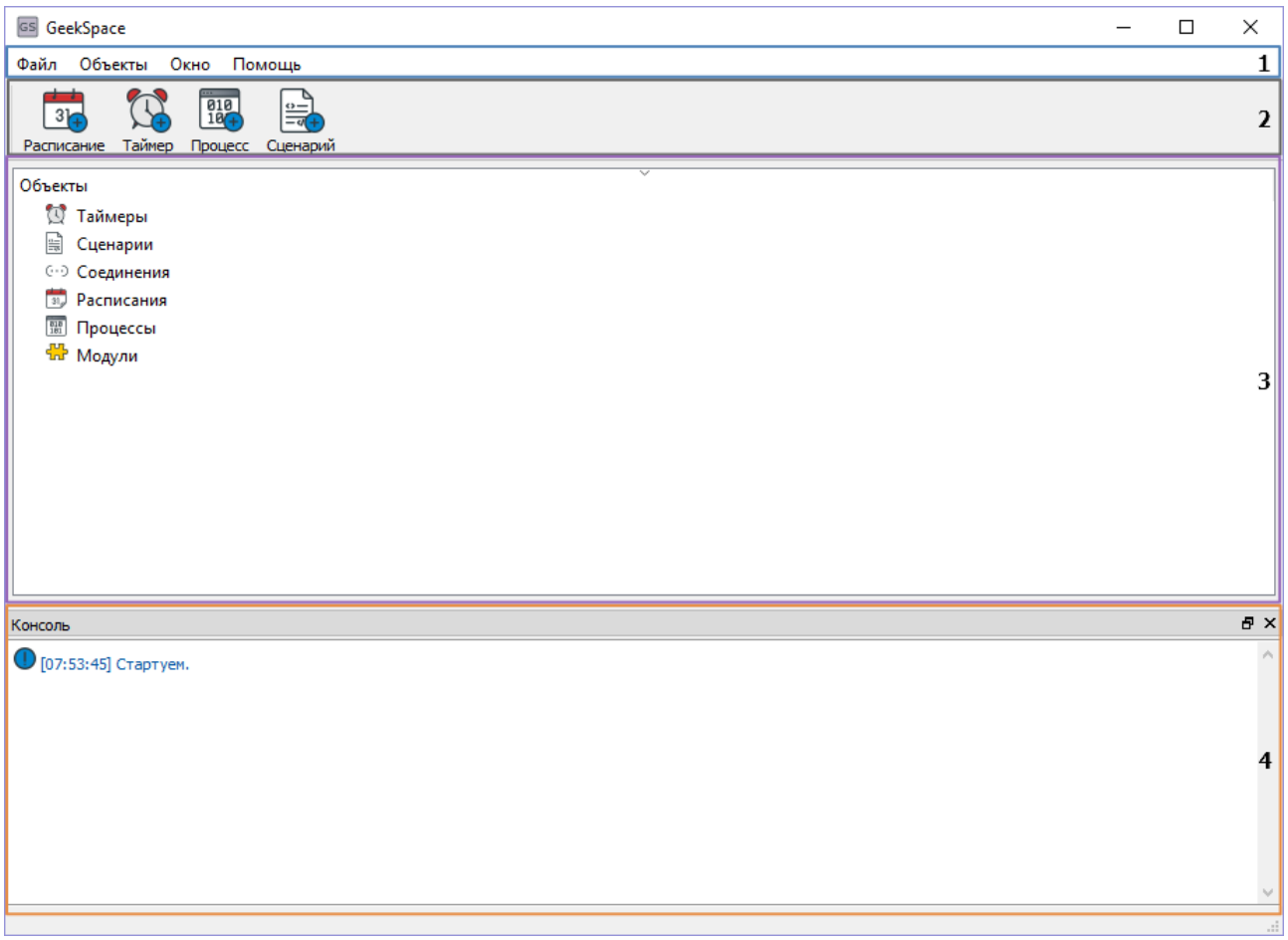

Рисунок 2.8.1 – Интерфейс программы.

Интерфейс программы разделён на 4 части:

1. Главное меню - содержит основные операции с объектами и интерфейсом;
2. Панель инструментов - включает в себя кнопки добавления объекта;
3. Область объектов - отображает список добавленных объектов и соединений;
4. Консоль - выводит основные сообщения о работе системы и её модулях.

Объекты

Для генерации дополнительных событий в ПО GeekSpace помимо модулей добавлены следующие объекты:

- Таймер - генерирует сигнал “timeout()” через выбранный промежуток времени и имеет возможность перезапуска бесконечно или заданное количество раз.
- Расписание. Сигнал “timeCome()” вызывается при наступлении выбранного даты и времени.
- Процесс. Позволяет запускать внешние приложения с необходимыми параметрами.
- Сценарий необходим для описания набора слотов для выполнения стандартных операций. Это позволяет упростить соединение сигналов со слотами.

Добавление объектов

Добавление объектов происходит через вызов главного меню, Объекты, Добавить и выбор необходимого для добавления объекта.

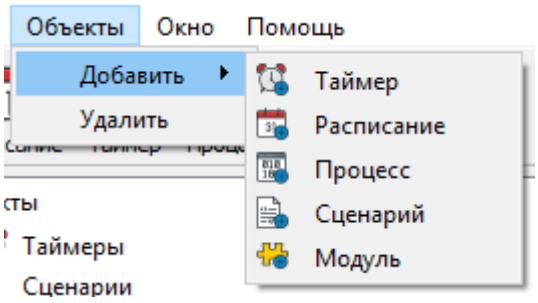

Рисунок 2.8.2 – Пункт Объекты главного меню .

Или через панель инструментов.

Для каждого объекта имеется собственный диалог для добавления.

Диалог добавления таймера

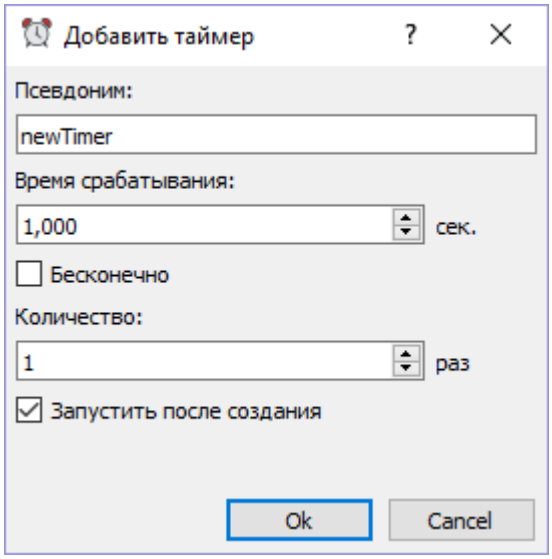

Рисунок 2.8.3 – Диалог “Добавить таймер”.

Псевдоним - название добавляемого объекта в системе, через которое будет происходить обращение к объекту.

Время срабатывания - период, через который будет происходить вызов сигнала “timeout()”

Флажок “Бесконечно” настраивает объект таймер на неограниченный повтор перезапуска таймера, однако остановить его можно через слот “stop()”.

Количество - параметр определяющий количество повторов таймера, после которого таймер будет остановлен.

Флажок “Запустить после создания” определяет будет ли запущен таймер после нажатия кнопки “Ок”.

Диалог добавления расписания

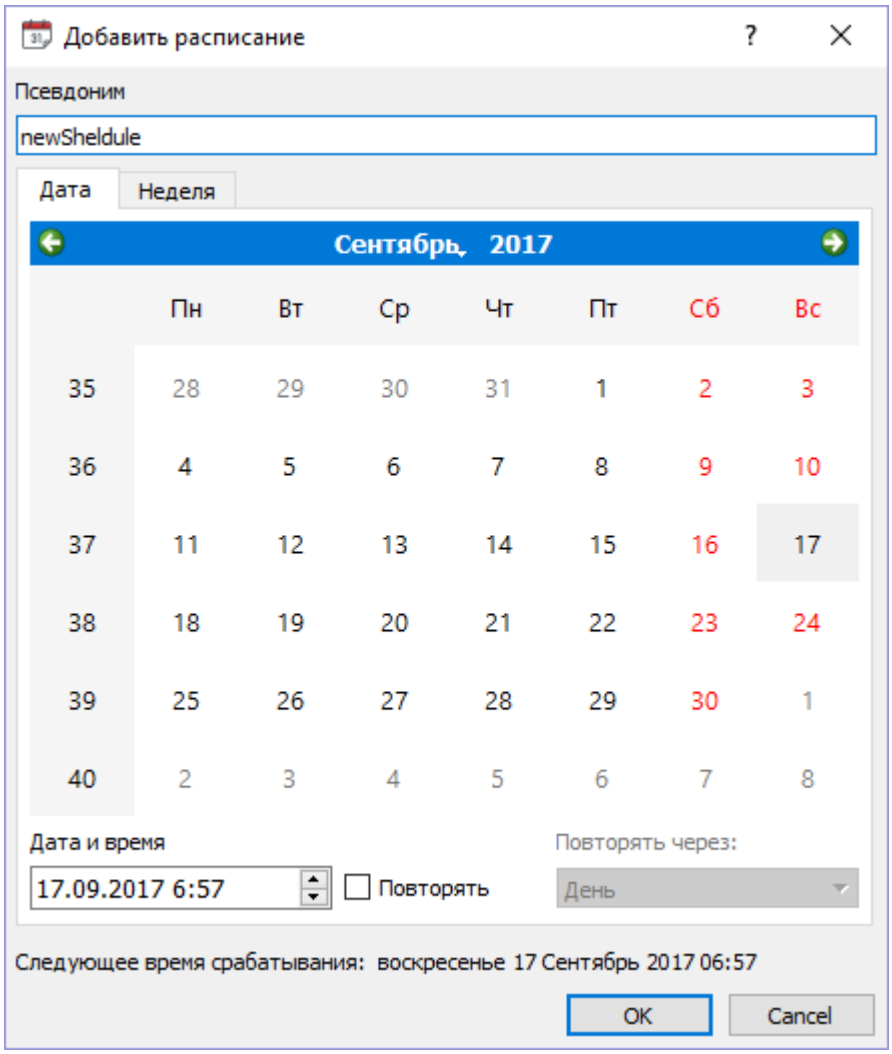

Рисунок 2.8.4 – Диалог “Добавить расписание”, вкладка “Дата”.

Данный диалог имеет две вкладки: “Дата” и “Неделя”.

На вкладке дата пользователь может выбрать нужную дату в области календаря или ввести её в поле ввода “Дата и время”.

Флажок “Повторять” определяет, необходим ли повтор расписания или же оно будет выполнено единожды.

Раскрывающиеся меню “Повторять через” позволяет пользователю настроить повтор расписания через определённые промежутки времени:

- День;
- Неделя;

- Месяц;
- Год.

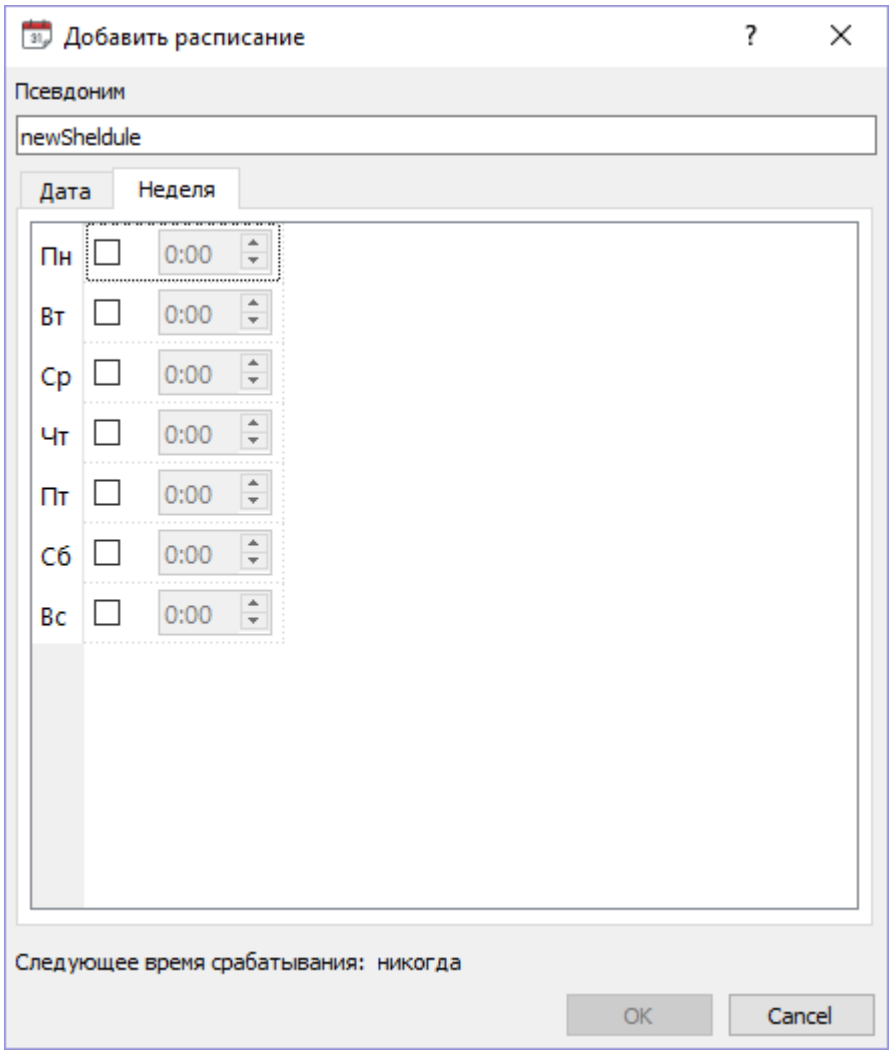

Рисунок 2.8.5 – Диалог “Добавить расписание”, вкладка “Неделя”.

На вкладке “Неделя” пользователь может настроить расписание срабатывания на каждый день недели. При этом на один день недели можно настроить сразу несколько значений времени. Новый столбец времени автоматически добавляется, если хотя бы один флажок в последнем столбце был отмечен.

Под вкладками отображается время следующего срабатывания сигнала “timeCome()”. Если дата указана ранее текущей даты, то срабатывание произведено не будет.

Диалог добавления процесса

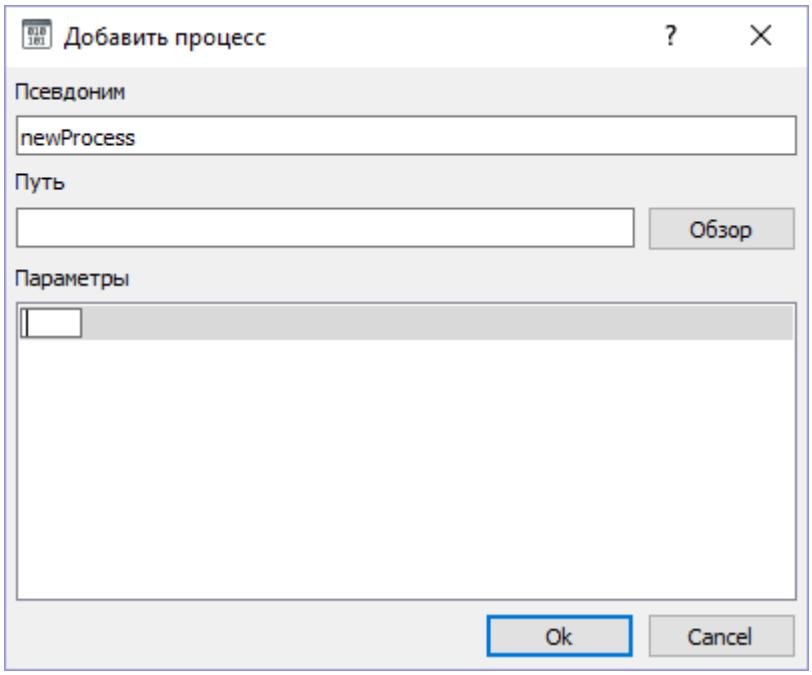

Рисунок 2.8.6 – Диалог “Добавить процесс”.

Путь - строка, указывающая на полный путь до исполняемого процесса.

Параметры - параметры запуска, которые передаются приложению при его запуске. Каждый новый параметр пишется на отдельной строке. Например, если необходимо в браузере Chrome открыть страницу в новом окне, то нужно указать два параметра, как на рисунке ниже.

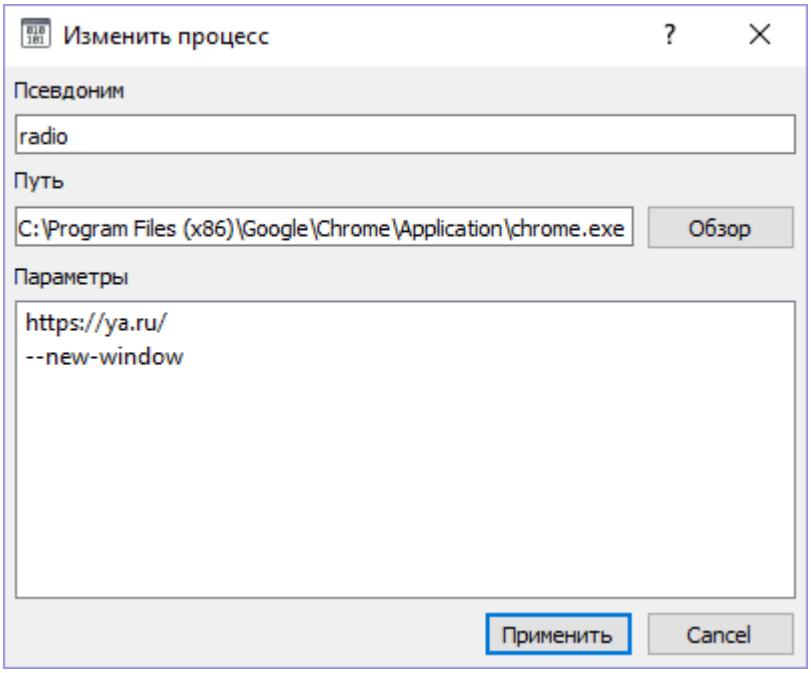

Рисунок 2.8.7 – Пример добавления процесса с несколькими параметрами.

Добавление сценария

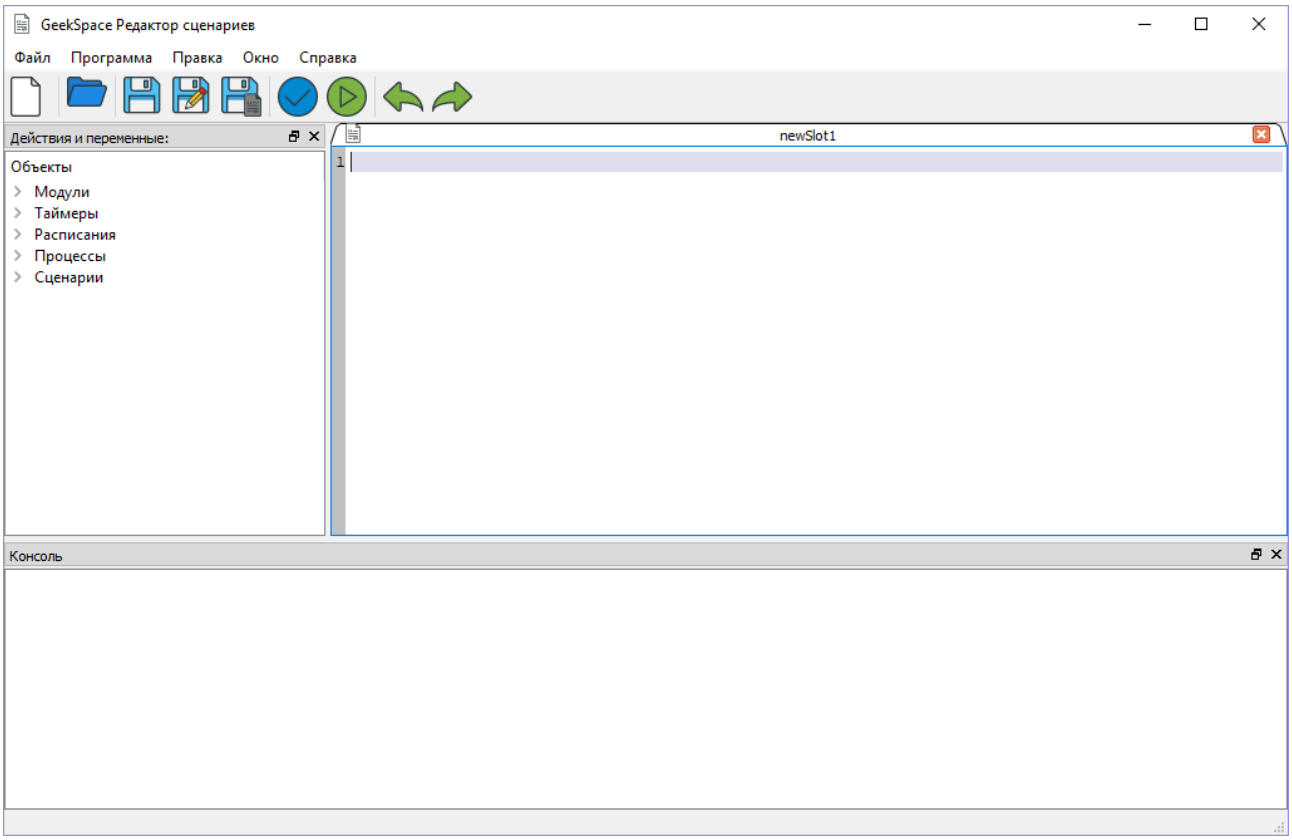

Рисунок 2.8.8 – Интерфейс редактора сценариев.

Процесс добавления сценария отличается от создания объекта, так как требует составления пользователем из уже существующих объектов. Так же необходима валидация составленного сценария. Поэтому в GeekSpace добавлен редактор сценариев, который позволяет проверить сценарий на правильность перед сохранением, а также его протестировать.

Код сценария можно добавлять как вручную, так и из панели “Действия и переменные”. При двойном клике на слот объекта он добавляется в текущую позицию курсора.

На панели инструментов редактора располагаются основные действия с текущим сценарием:

- Создать новый;
- Открыть из файла;
- Сохранить;
- Сохранить как;
- Сохранить как черновик (в файл);
- Проверить сценарий;
- Протестировать (проверить и выполнить сценарий);
- Отменить последнее действие;
- Повторить отменённое действие;

Добавление модуля

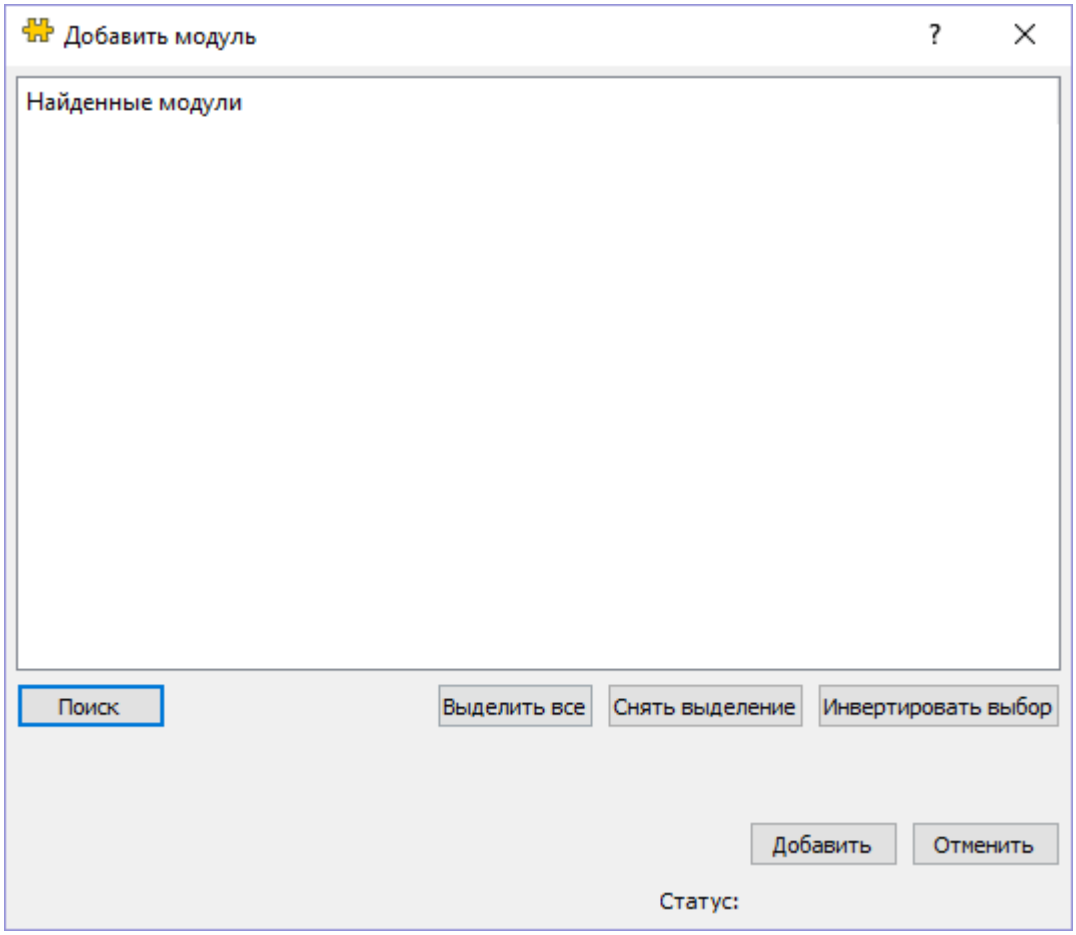

Рисунок 2.8.9 – Диалог “Добавить модуль”.

В систему GeekSpace встроен автоматический поиск модулей в локальной сети. Для этого необходимо в диалоговом окне нажать кнопку поиск. Когда поиск будет окончен, в нижней части экрана появится надпись “Поиск завершен”, а все доступные модули, но не добавленные в систему отобразятся в древовидном списке.

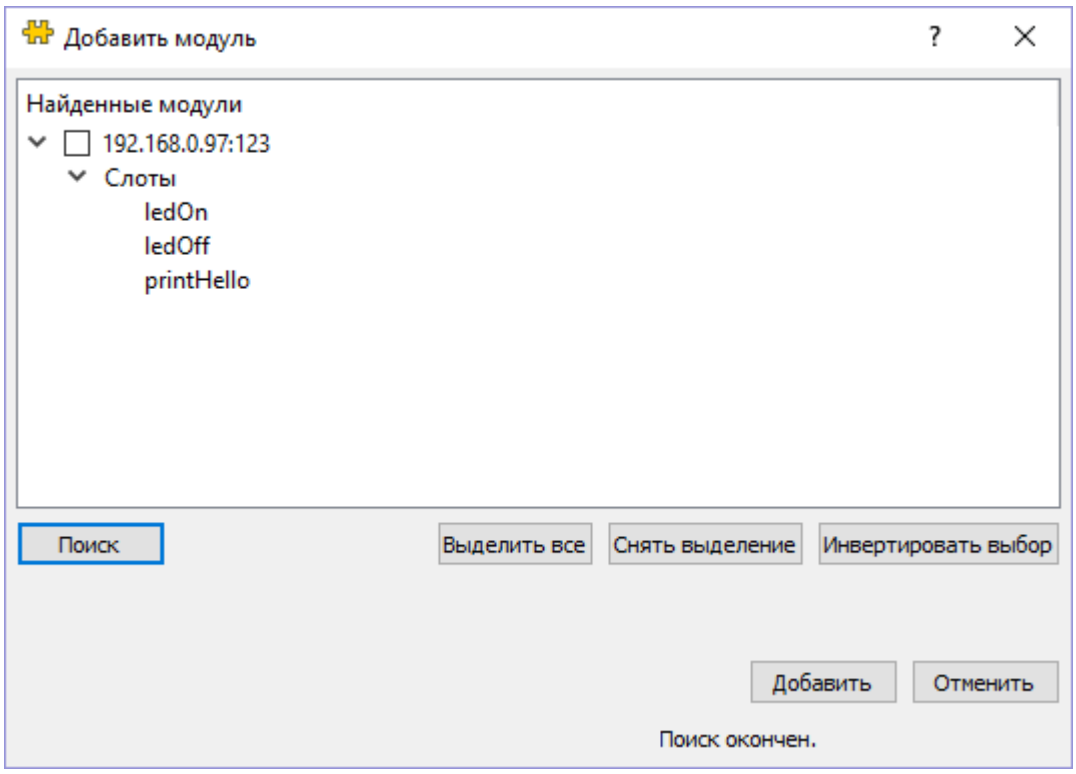

Рисунок 2.8.10 – Пример найденного модуля в локальной сети.

Пользователь может выбрать необходимые модули, отметив их галочкой. И после нажатия кнопки “Добавить” все отмеченные модули будут добавлены в систему.

Удаление объектов

Удаление объектов происходит через контекстное меню и панель инструментов.

Для удаления через контекстное меню нужно нажать правой кнопкой мыши по удаляемому объекту и выбрать пункт меню “Удалить объект”. Пример контекстного меню на рисунке ниже.

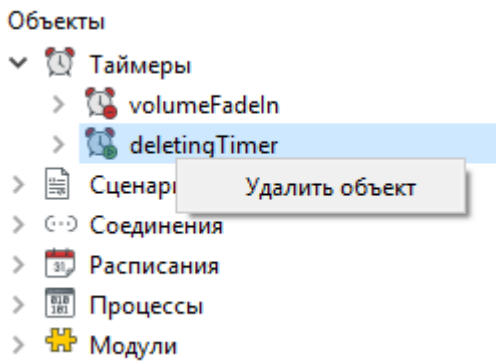

Рисунок 2.8.11 – Контекстное меню объекта.

Чтобы удалить объект через главное меню, нужно выделить удаляемый объект и перейти по: главное меню, Удалить.

Добавление связей

Создание связи сигнал - слот происходит через контекстное меню сигнала. Необходимо раскрыть ветку сигналов объекта и кликнуть правой кнопкой по нужному сигналу объекта. Во всплывающем меню выбрать слот который требуется вызвать по сигналу.

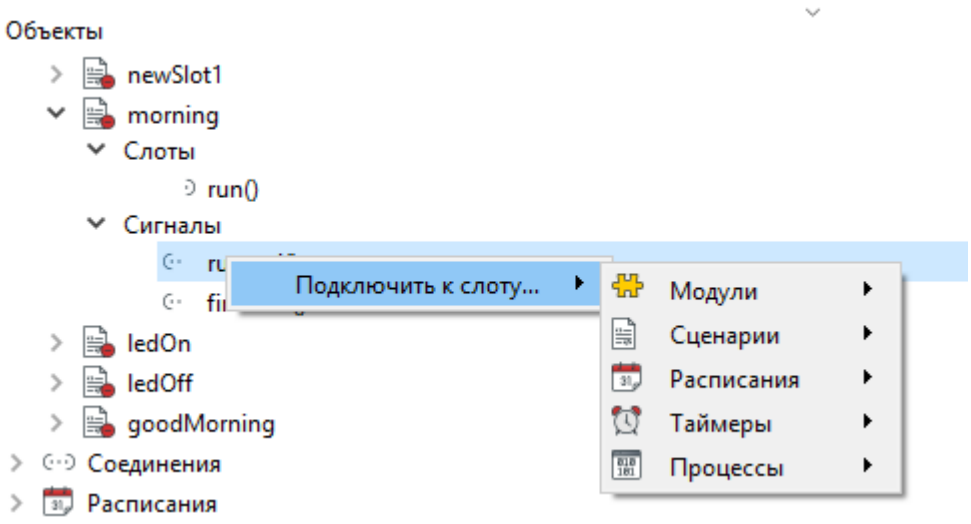

Рисунок 2.8.12 – Пример контекстного меню сигнала объекта.

Все связанные сигналы и слоты отображаются в ветке “Соединения” и сгруппированы по вызывающим сигналам.

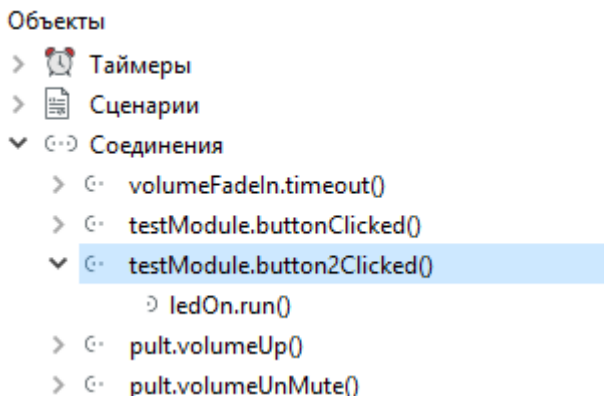

Рисунок 2.8.13 – Пример ветки “Соединения”.

Удаление связей

Чтобы удалить связь нужно вызвать контекстное меню слота в ветке “Соединения” и выбрать во всплывающем меню “Удалить соединение”.

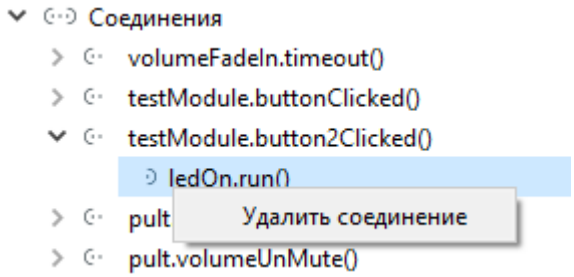

Рисунок 2.8.14 – Контекстное меню слота соединения.

2.9. Тестирование программной системы

Для тестирования и отладки программной системы была реализована упрощённая конфигурация со схожими требуемыми функциями. На модуле “light”, как аналоге вентиляции, в качестве устройства управления был использован источник света с несколькими уровнями яркости в зависимости от уровня освещённости. На нём были описаны следующие функции-слоты:

- led100 - включить источник света на полную мощность;
- led75 - включить источник света на 75% мощности;
- led25 - включить источник света на 25% мощности;

- led0 - выключить источник света;
- checkLight - проверка уровня освещённости на датчике и вызов соответствующего сигнала;
- ledUp - увеличить уровень света на один уровень;
- ledDown - уменьшить уровень света на один уровень.

И для обеспечения связи ПО и модуля были добавлены сигналы:

- lightLow - низкий уровень освещённости;
- lightBelowMiddle - уровень освещённости ниже среднего;
- lightAboveMiddle - уровень освещённости выше среднего;
- lightHight - высокий уровень освещённости.

Чтобы организовать автоматизацию регулирования были соединены сигналы об уровнях освещённости со слотами устанавливающим необходимый уровень источника света. Затем создан таймер с желаемым интервалом обновления и подключен его сигнал timeOut к слоту checkLight. Набор соединений сигнал => слот:

- lightCheckTimer.timeout => light.checkLight
- light.lightHight => light.led0
- light.lightBelowMiddle => light.led75
- light.lightAboveMiddle => light.led25
- light.lightLow => light.led100

На модуле “locker”, который имитирует работу устройства доступа, были установлены две кнопки имитирующих правильный и неправильный ввод кода доступа и сервопривод как устройство управления открыванием и закрыванием замка. Далее для расширения функционала установлен инфракрасный приёмник, который способен принимать сигнал от пультов дистанционного управления работающих на инфракрасной связи. А также светодиод-индикатор указывающий о состоянии замка для простоты визуального контроля.

Реализованы следующие функции-слоты:

- Close - имитация закрытия;
- Open - имитация открытия.

Сигналы управления замком:

- buttonValid - нажатие кнопка, имитирующая правильный ввод пароля;
- buttonInvalid - нажатие кнопка, имитирующая не правильный ввод пароля;

И сигналы инфракрасного пульта соответствующих кнопок:

- Plus
- Minus
- Mute
- UnMute
- Up
- Down
- arrowRight
- arrowLeft
- arrowUp
- arrowDown
- off

Для того, чтобы обеспечить не только управление доступом, но и его контроль необходимо было создать два объекта процессов отвечающих за запись в лог-файл данных об успешных и неуспешных попытках доступа, как показано на рисунке ниже.

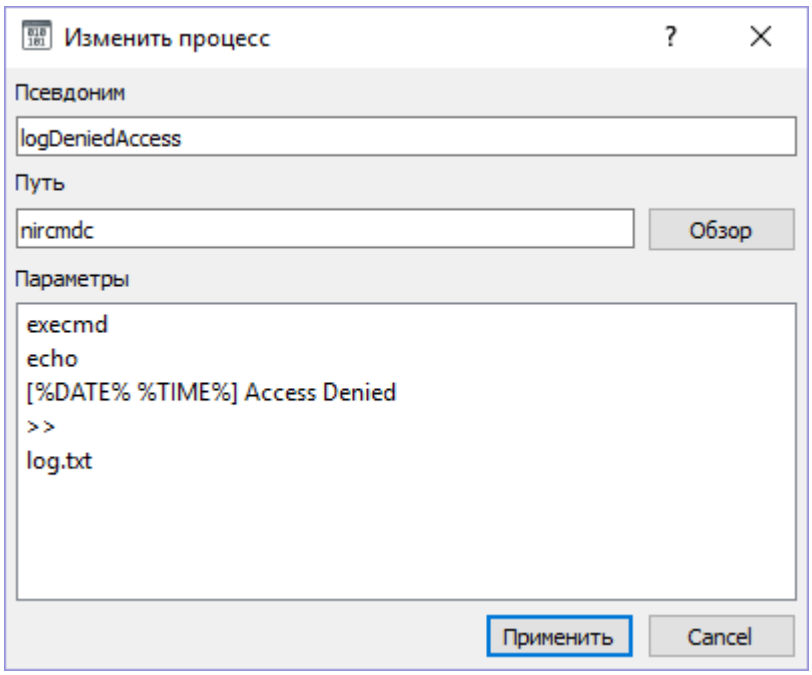

Рисунок 2.9.1 – Настройки объекта процесс, для записи в лог-файл данных о неудачной попытке входа.

И было решено добавить звуковое сообщение об удачном и неудачном входе в систему.

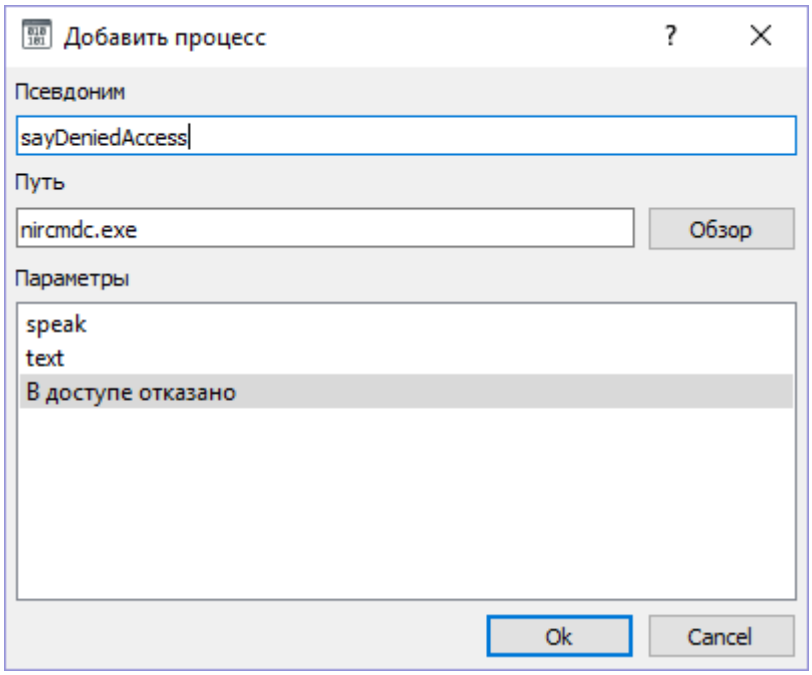

Рисунок 2.9.2 – Настройки объекта процесс, для звукового оповещения о неудачной попытке входа.

Для того, чтобы дверь могла автоматически закрываться после входа человека в помещение был добавлен объект таймера “openLockerTimer” установленный на 15 секунд. И было необходимо сделать соединение слота timeOut со слотом закрывания двери:

openLockerTimer.timeout => locker.Close

Чтобы не производить большое количество подключений были созданы два объекта сценария, в которых описано поведение при удачной и неудачной попытке входа.

Текст сценария “accessGranted” для удачной попытки:

```
logAccessGranted.start();
speakAccessGranted.start();
locker.Open();
openLockerTimer.start();
```

Текст сценария “accessDenied” для неудачной попытки:

```
logAccessDenied.start();
speakAccessDenied.start();
```

Чтобы обрабатывать попытки входа сигналы модуля “locker” были соединены с соответствующими слотами сценариев:

locker.buttonValid => accessGranted.run

locker.buttonInvalid => accessDenied.run

В завершении был добавлен функционал управления светом с инфракрасного пульта дистанционного управления. Для этого потребовалось создать два сценария, которые отключают автоматическую настройку света и вызывают слот изменения яркости освещения.

Сценарий “lightUp” для ручного увеличения яркости освещения:

```
lightCheckTimer.stop();
light.ledUp();
```

Сценарий “lightDown” для ручного уменьшения яркости освещения:

```
lightCheckTimer.stop();
light.ledDown();
```

Соединения для управления светом:

locker.Up => lightUp.run

locker.Down => lightDown.run

locker.arrowRight => lightCheckTimer.start

В завершении были добавлены два расписания

- “lightControllStop” - останавливающее автоматическую регулировку света и выключающий его;
- “lightControllStart” - запускающее автоматическую регулировку света.

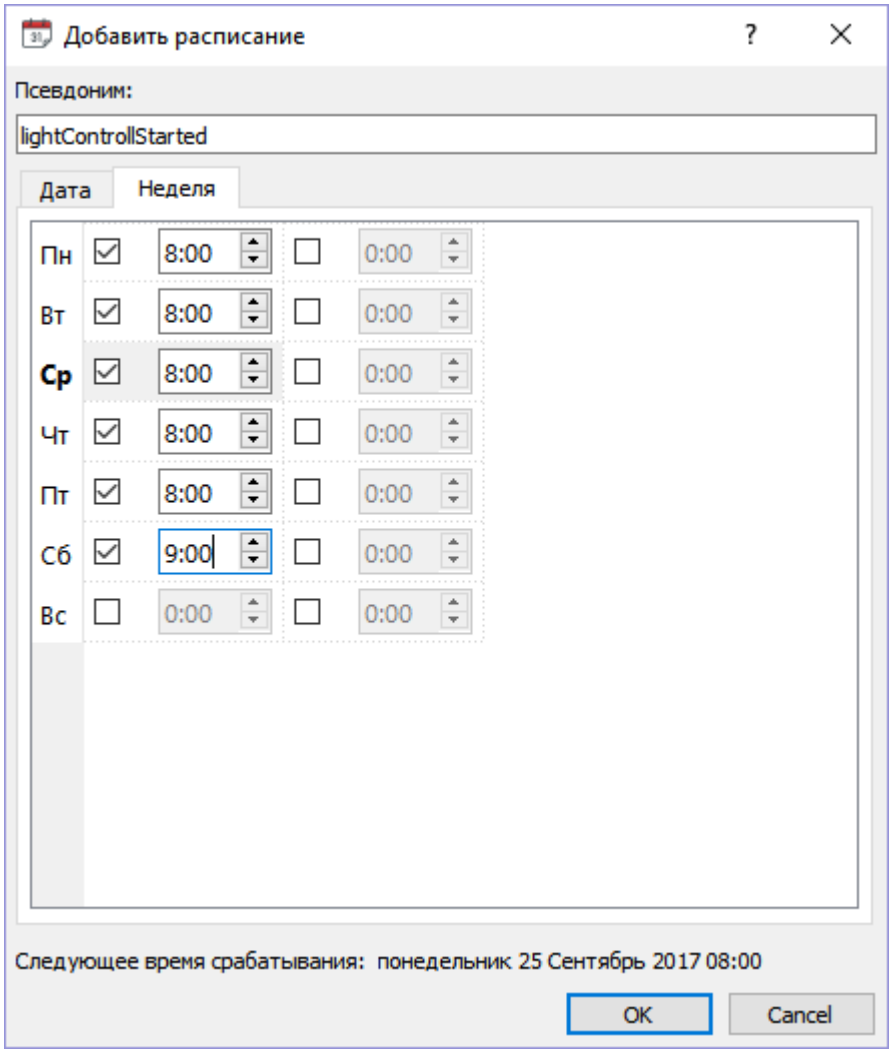

Рисунок 2.9.3 – Настройка расписания включения света.

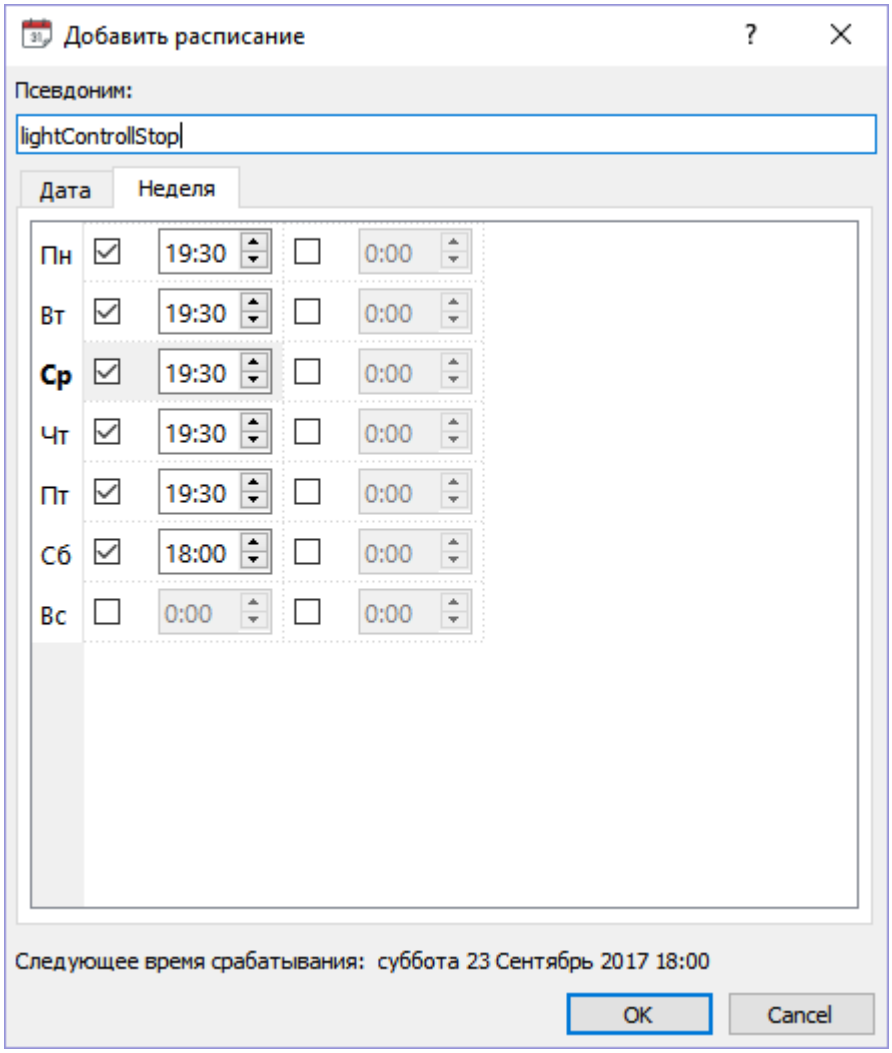

Рисунок 2.9.4 – Настройка расписания выключения света.

И установлены соединения, как показано ниже:

lightControllStop.timeCome => light.led0

lightControllStop.timeCome => lightCheckTimer.stop

lightControllStart.timeCome => lightCheckTimer.start

В итоге для тестирования программной системы были созданы следующие объекты:

- Модули:
  - light - для управления системой освещения;
  - locker - для управления доступом и управления системой с ИК пульта.
- Таймеры:

    - lightCheckTimer - для проверки уровня освещённости;
    - openLockerTimer - для задержки перед закрывание двери.
- Процессы:
    - speakAccessDenied - для звукового сопровождения успешного доступа;
    - speakAccessGranted - для звукового сопровождения отказа в доступе;
    - logAccessGranted - для логирования удачного доступа;
    - logAccessDenied - для логирования отказа в доступе.
- Сценарии:
    - accessGranted - вызов слотов при успешном входе;
    - accessDenied - вызов слотов при отказе во входе;
    - lightDown - понижение уровня освещённости и отключение автоматической регулировки;
    - lightUp - увеличение уровня освещённости и отключение автоматического регулирования.
- Расписания:
    - lightControllStop - остановка регулирования и выключение света;
    - lightControllStart - запуск автоматического регулирования света;
- Соединения:
    - lightCheckTimer.timeout = light.checkLight;
    - light.lightHight = light.led0;
    - light.lightBelowMiddle = light.led75;
    - light.lightAboveMiddle = light.led25;
    - light.lightLow = light.led100;
    - locker.arrowRight = lightCheckTimer.start;
    - openLockerTimer.timeout = locker.Close;
    - locker.Up = lightUp.run;

- locker.Down =lightDown.run;
- locker.buttonValid = accessGranted.run;
- locker.buttonInvalid = accessDenied.run;
- lightControllStop.timeCome = lightCheckTimer.stop;
- lightControllStop.timeCome = light.led0;
- lightControllStart.timeCome = lightCheckTimer.start.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе были достигнуты следующие цели: создана кроссплатформенная программная система состоящая из интерфейсного программного обеспечения GeekSpace и библиотеки GeekSpace Module для описания и настройки автоматизации системами жизнеобеспечения помещения и их управления; успешно проведена апробация системы на тестовом стенде имитирующем аналогичные системы жизнеобеспечения; исходный код библиотеки и ПО были выложены в общий доступ, в сервисе для хостинга IT-проектов GitHub. Это позволит людям связанных с около компьютерными технологиями использовать данную программную систему.

Дальнейшей перспективой развития программной системы может служить разработка следующего функционала:

- Алгоритмы шифрования и дешифрования для безопасного обмена данными между ПО GeekSpace и её модулями;
- Слоты и сигналы с входящими и исходящими параметрами;
- Связь с базами данных и выполнение запросов к ним;
- Переменные объектов;
- Полноценный си-подобный скриптовой язык для описания взаимодействия между модулями;
- Виртуальная машина для выполнения скриптов;
- Создание библиотеки для мобильных приложений, которая позволит включить смартфон на различных операционных системах в состав системы GeekSpace в качестве модуля.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Welcome to the openHAB 2 Documentation! [Электронный ресурс]: Документация к проекту openHAB. - Режим доступа: http://docs.openhab.org/ - Загл. с экрана.
2. OpenHAB — стань программистом собственного жилища. [Электронный ресурс]: Хабрахабр – самое крупное в Рунете сообщество людей, занятых в индустрии высоких технологий. - Режим доступа: https://habrahabr.ru/post/232969/ - Загл. с экрана.
3. Программируем управление освещением по датчикам движения и освещения на Node-RED. [Электронный ресурс]: Geektimes – самое крупное в Рунете сообщество людей, занятых в индустрии высоких технологий. - Режим доступа: https://geektimes.ru/post/279814/ - Загл. с экрана.
4. Компьютерные сети. Принципы, технологии, протоколы. / Олифер В.Г., Олифер Н.А. – СПБ: Питер, 2016. – 992 с.
5. Сигналы и слоты в Qt [Электронный ресурс]: Хабрахабр – самое крупное в Рунете сообщество людей, занятых в индустрии высоких технологий. - Режим доступа: https://habrahabr.ru/post/50812/ - Загл. с экрана.
6. NirCmd - Windows command line tool [Электронный ресурс]: NirSoft web site provides a unique collection of small and useful freeware utilities, all of them developed by Nir Sofer. - Режим доступа: http://www.nirsoft.net/utils/nircmd.html - Загл. с экрана.
7. Auto Clicker [Электронный ресурс]: Dual Monitor Software, Auto Clicker, Mouse Cursor Automation Utilities and more - Режим доступа: https://www.murgee.com/auto-clicker/ - Загл. с экрана.